Precision Angle Seeking in Robots: A Reinforcement Learning Approach in Simulation and Reality

Anonymous Author(s)
Affiliation
Address
email

Abstract: This paper explores the deployment and challenges of Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) for precision angle seeking in robotic control across both simulated and physical environments. We highlight the transition from theory to application by introducing the Angular Positioning Seeker (APS) environment, leveraging Raspberry Pi 4B+ platforms to rigorously evaluate RL algorithms in scenarios that closely mimic real-world conditions. This benchmark is designed to shed light on the subtleties distinguishing RL implementation in physical realms from simulated proxies, thereby fostering advancements and more nuanced testing protocols within the domain of robotic intelligence. The source code related to this paper is available at https://github.com/RoboAngleSeeker/RoboAngleSeeker.

Keywords: Reinforcement Learning in Robotics, Simulation-to-Reality Transfer, Deep Reinforcement Learning Algorithms

1 Introduction

The application of Reinforcement Learning (RL) in robotics is a frontier in artificial intelligence, promising significant automation benefits. **However, understanding the challenges of applying RL in real-world scenarios is crucial.** Translating RL from theory to practice requires rigorous empirical validation to ensure reliable performance in varied environments.

Exploring the complexity of RL applications reveals significant challenges. A synthesis of the state-of-the-art in RL policy representation in robotics underscores the intricacies involved, with solutions such as Gaussian Mixture Models and Dynamic Movement Primitives proposed by Kormushev et al. [1]. These challenges are further highlighted by examples like pancake flipping and energy-efficient bipedal walking [2]. Integrating action planning with RL for solving robotic causal problems in noisy environments also demonstrates the broad applicability and versatility of RL [3]. Applying these advanced techniques in practice involves complex technical details [7, 8, 9], showcasing the need for thorough understanding and precise implementation.

One critical issue in RL deployment is managing uncertainty and its calibration. Addressing uncertainties is vital to ensure robust performance, as highlighted by various studies [10, 11, 12, 13]. Effective calibration of uncertainty is equally challenging and essential [14, 15, 16, 17, 18]. These challenges underscore the complexity of deploying RL in unpredictable real-world environments.

Transitioning from simulation to real-world applications introduces additional challenges. Despite progress in DRL for robust control, issues like sampling efficiency and safety remain prominent. Ju et al. [19] explore methodologies inspired by transfer learning to bridge this gap, such as feature commonality extraction and multitasking. Benchmarks like RMBench [20] are crucial for evaluating RL algorithms in robotic manipulation tasks. Advances in DRL for robotic manipulation,

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

focusing on sample efficiency and generalization, have been documented by Liu et al. [21]. Other frameworks, such as those presented by Caponio et al. [22] and Wu et al. [23], further highlight the diverse applications and ongoing developments in this field. Additionally, Margolis et al. and Orr et al. [24, 25] discuss ADP/RL in various control systems. Han et al., Bhagat et al., and Osband et al. [26, 27, 28] review recent DRL developments in robotic manipulation, identifying issues and solutions. Yang et al. [29] explore advancements in robot affordance learning for sensory-based task performance. Advanced reinforcement learning techniques can be used for learning the manipulation of deformable objects by robots [30, 31, 32], with task migration being another important direction [33, 34, 35].

Effective innovative approaches, such as generative language models and human-in-the-loop interventions, are essential for advancing reinforcement learning algorithms. Dastider et al. [36] unveil a motion planning framework using manifold learning and variational autoencoding, outperforming traditional methods in efficiency. Enhancements in robot performance through human language feedback have been demonstrated by Lucy Xiaoyang Shi et al. [37]. Real-time human intervention in imitation learning is also a common approach [38, 39, 40, 41, 42, 43]. The use of large-scale models like GPT4-V for mapping environmental inputs to robot actions is paving the way for future RL directions [44, 45], with specialized datasets emerging for training robots in action learning [46, 47, 48, 49].

Despite the progress, there is a lack of effective benchmarks for experimental validation. Our work addresses this by proposing a benchmark using Raspberry Pi 4B+ platforms for simulating and physically testing RL algorithms, demonstrating promising results. This benchmark provides a deeper understanding of the differences between RL in simulation and real-world environments, aiding more effective testing for the research community.

2 Method

We created the Angular Positioning Seeker (APS) environment in OpenAI's Gym, tailored for angular positioning tasks using ROS.

2.1 The Angular Positioning Seeker

We created the Angular Positioning Seeker (APS) environment in OpenAI's Gym, tailored for angular positioning tasks using ROS. The step function logic was restructured to prioritize angle-seeking behavior. For a given action a, representing the motor's torque, the updated angular velocity $\dot{\theta}_{new}$ is determined by:

$$\dot{\theta}_{new} = \dot{\theta} + \left(-\frac{3g}{2l} \sin(\theta + \pi) + \frac{3}{ml^2} a \right) \Delta t \tag{1}$$

Here, g stands for the acceleration due to gravity, l is the length of the arm, m is the mass, θ represents the current deviation from the target angle, $\dot{\theta}$ is the angular velocity, a is the applied torque, and Δt is the time increment.

The angle θ_{new} is updated considering the target position as the reference:

$$\theta_{new} = \theta + \dot{\theta}_{new} \Delta t \tag{2}$$

The reward function was redesigned to minimize deviation from the target angle:

$$\text{costs} = (\theta - \theta_{target})^2 + 0.1\dot{\theta}^2 + 0.001a^2 \tag{3}$$

where θ_{target} is $\frac{\pi}{2}$ radians for 90 degrees.

These modifications collectively adapt the original pendulum's dynamics to fit our unique Angular Positioning Seeker's requirements, facilitating research into angle-seeking strategies.

2.2 Apply DQN on APS

We employed the DQN algorithm and its variants to simulate and analyze their performance in the inverted APS environment.

In the context of employing Deep Q-Network (DQN) and Double DQN methodologies for the physical realization within an APS environment, the following pseudocode encapsulates the core algorithmic structure adapted for this specific application. The state inputs are derived from an MPU6050 three-axis accelerometer and gyroscope sensor, while the network architectures faithfully replicate those of DQN and Double DQN. The action space is tailored to the operational dynamics of a DC motor, which serves as the actuator in this setup.

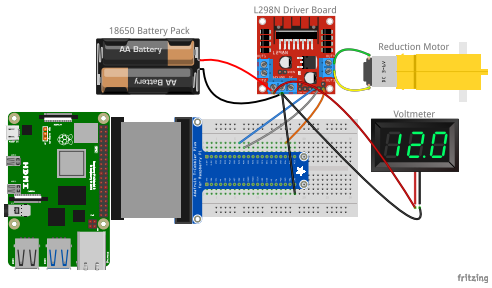
Algorithm 1 DQN and Double DQN for Physical APS Control with Raspberry Pi

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
Initialize Raspberry Pi GPIO pins for motor and MPU6050 sensor
Calibrate MPU6050 and define PWM signal mapping for motor control
Define action-to-velocity mapping $v : \{0, \ldots, 10\} \mapsto \{-2, \ldots, 2\}$
for episode $= 1$ to M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for $t = 1$ to T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q(\phi(s_t), a; \theta)$
    Convert action a_t to motor velocity v_t using mapping v
    Apply v_t to motor via PWM adjustment
    Observe reward r_t and new state x_{t+1} from MPU6050
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    if episode terminates at step $j + 1$ **then**
      Set $y_j = r_j$
    else
      Set $y_j = r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-)$
    end if
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to θ
    Every C steps reset $\hat{Q} = Q$
  end for
end for

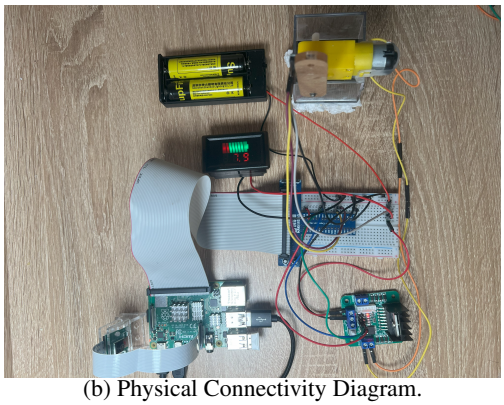
2.3 Building a Physical Setup

To construct the physical apparatus, we employed a Raspberry Pi 4B+ in conjunction with an STM32F103ZET6 microcontroller.The circuit diagram for the Raspberry Pi version of the inverted APS setup is shown in Figure 1a.

The physical connectivity of the setup is illustrated in the following diagram. In our experimental setup, various components are utilized for controlling and monitoring purposes. The detailed descriptions of these components can be found in Table 1. The Raspberry Pi 3 mainboard serves as the central control unit, communicating with the L298N module through the GPIO interface. Additionally, the power supply for the Raspberry Pi, the L298N module for driving DC motors, and other essential components such as the DC motor and voltmeter are included in our setup. These components collectively facilitate the execution and monitoring of our experiments. For more information on building physical environments, please refer to the appendix.



(a) Circuit Diagram with Pi4B+.



(b) Physical Connectivity Diagram.

Figure 1: **(a):** Circuit Configuration for Raspberry Pi 4B+ in Inverted APS Control Experiment. This schematic represents the core electrical connections, featuring a Raspberry Pi 4B+ and STM32F103ZET6 microcontroller, which facilitate the implementation of reinforcement learning algorithms in a physical APS setup. **(b):** Physical Connectivity of Inverted APS Components. The diagram showcases the integration of the Raspberry Pi 3 mainboard with other essential hardware components, such as the L298N motor drive module and DC motors, which are pivotal for the real-world application of DQN algorithms in APS stabilization.

Table 1: Components and Descriptions

Component	**Description**
Raspberry Pi 3 mainboard	Runs control program, communicates with L298N module via GPIO interface.
Raspberry Pi power supply	Provides stable power supply for the Raspberry Pi.
L298N module	Drives DC motors, accepts control signals from Raspberry Pi, controls motor direction and speed.
DC3V-6V geared DC motor TT motor	DC motor driven by L298N module in the experiment.
Mini digital DC voltmeter	Monitors battery voltage to ensure stable power supply.
18650 rechargeable lithium batteries	Provides power for L298N module and DC motors.
Breadboard	Used for circuit construction and testing.
40P flexible flat cable	Connects Raspberry Pi and L298N module, transmits control signals.
Jumper wires	Used for circuit connections, ensuring circuit continuity.

3 Evaluation

The evaluation focuses on the algorithmic performance, showcasing the results through various metrics and visualizations.

3.1 Inverted APS Performance Results

We evaluated DQN, Double DQN, and Dueling DQN on the APS environment, using return and Q-value metrics to illustrate learning efficiency and algorithm effectiveness.

As depicted in Figure 2 (a) and (d), this figure illustrates the disparity in performance of the DQN algorithm between simulated and real-world settings. The performance of DQN exhibits a notable similarity across both environments. However, a significant drop in rewards is observed in the simulation at episode 65, plunging from -150 to -700 before swiftly reverting to normal levels, thereby demonstrating commendable stability. In contrast, fluctuations nearing convergence in the real environment are considerably more pronounced than in the simulation, with a fluctuation range of approximately 150 to 200. Nonetheless, convergence around a reward of 200 is eventually achieved, with the video on our website (accessible through our open-source repository) showcasing its robust performance.

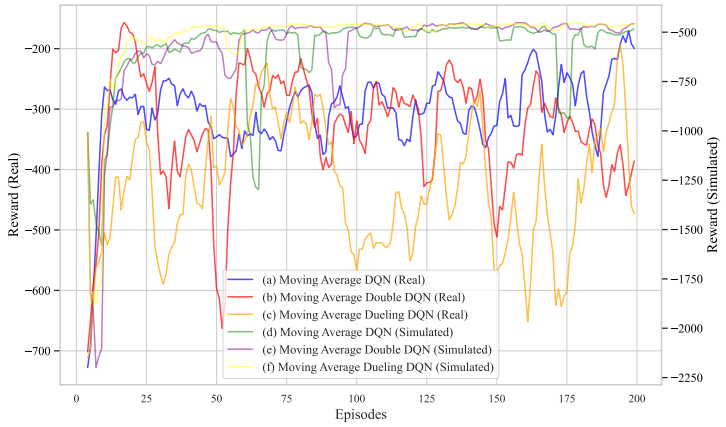


Figure 2: Differential Performance of Reinforcement Learning Algorithms in Real and Simulated Environments. (1) DQN showcases a convergence around a reward of 200 in real environments, despite wider fluctuations, and a transient drop to -700 in simulations at episode 65. (2) Double DQN achieves a stable learning curve in simulations, while real-world performance demonstrates higher reward peaks followed by larger oscillations. (3) Dueling DQN rapidly attains high convergence in simulations, with real-world trials displaying more pronounced and frequent reward fluctuations.

Figure 3 (a) and (d) present the evolution of Q-values for the DQN algorithm in both simulated and real-world scenarios during the inverted APS task. A higher Q-value signifies the system's prediction that a particular action will lead to a more stable or closer-to-objective state (e.g., keeping the APS upright). In the simulation, Q-values for DQN converge more swiftly and remain within a narrow oscillation range, despite occasional sharp fluctuations, indicating moments of significant uncertainty with oscillations reaching up to 150. Conversely, in the real environment, Q-values change gradually, displaying a coherent trend of decrement until convergence. The convergence point of Q-values in the real environment is nearly identical to that in the simulation, settling around -30. This indicates a closely mirrored estimation of action values by DQN in both settings, underscoring the algorithm's capacity to adapt its policy towards achieving the target state across diverse environments.

As illustrated in Figure 2 (b) and (e), Double DQN demonstrates enhanced stability and smoother convergence in simulations compared to DQN, avoiding episodic reward plunges. However, in real-world scenarios, Double DQN initially secures higher rewards but subsequently faces significant drops, exhibiting greater oscillation amplitude and slightly lower convergence points than DQN. fig. 3 (b) and (e) show that Double DQN's Q-values have reduced fluctuations in simulations, improving predictability. Yet, in real-world settings, Double DQN's Q-values converge around -50, lagging behind DQN's -30, indicating that DQN outperforms Double DQN in accurately predicting outcomes for APS stabilization in real-world applications.

Figure 2 (c) and (f) reveal that Dueling DQN excels in simulations, achieving high convergence values with notable stability and minimal fluctuations. However, in real-world settings, it lacks clear convergence and exhibits broad oscillations between -600 and -200. Figure 3 (c) and (f) show that Dueling DQN has the least Q-value variability in simulations, with fluctuation spikes around 20, indicating superior predictability. In contrast, in real environments, its Q-values hover around -50, similar to Double DQN's, and are inferior to DQN's, which converge around -30. This suggests that DQN outperforms Dueling DQN in real-world applications, demonstrating greater determinism and stability.

3.2 Result Analysis

Examining DQN, Double DQN, and Dueling DQN revealed key insights into their adaptability. DQN showed superior real-world performance, attributed to its simpler structure that allows rapid

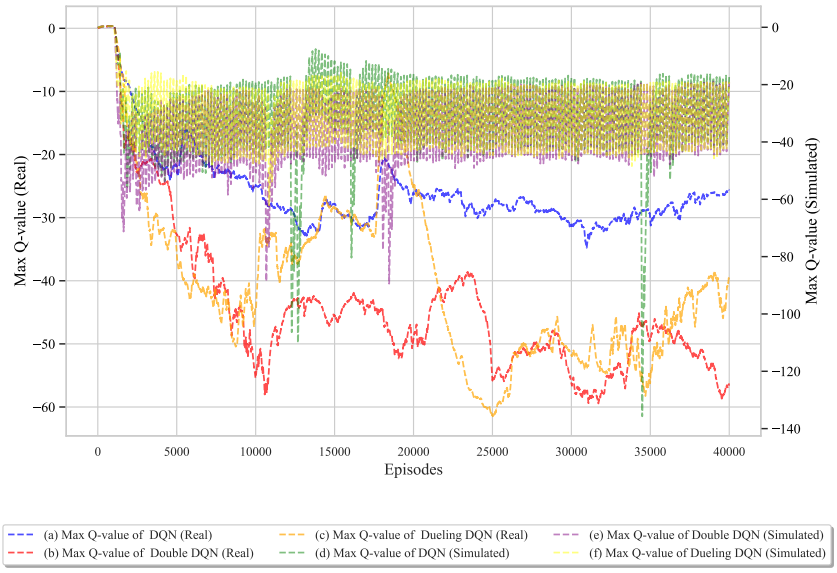


Figure 3: Trajectories of Q-value Convergence Across Environments for Reinforcement Learning Algorithms. (a) & (d) DQN shows swift convergence in simulations with Q-value oscillations up to 150, whereas real-world convergence is gradual with values settling around -30, demonstrating adaptability. (b) & (e) Double DQN offers reduced simulation oscillations with spikes decreasing to 60, but underperforms in real settings with Q-values converging around -50, indicating less predictability. (c) & (f) Dueling DQN maintains the lowest variability in simulations with spikes near 20 yet converges around -50 in real-world settings, paralleling Double DQN and revealing room for improvement in adaptability.

policy adjustments, indicating that algorithmic complexity does not necessarily improve performance in complex scenarios. This reinforces the idea that simplicity can enhance resilience against environmental uncertainties. Conversely, Dueling DQN demonstrated commendable stability in simulations due to its architectural design, which bifurcates state value and advantage functions. This design enables effective learning of state representations and steady value estimation, suggesting a potential direction for developing robust RL algorithms capable of discerning subtle differences in action outcomes with precision.

Figure 4 illustrates the action space exploration of DQN, Double DQN, and Dueling DQN, showing their alignment with fitted normal distributions. The proportions of the histogram's area outside the normal distribution curves for DQN, Double DQN, and Dueling DQN are 0.3237, 0.3563, and 0.1648, respectively, indicating Dueling DQN's superior action selection efficacy. Additionally, Figure 5a shows DQN's action variability with consistent oscillations, Double DQN's minor fluctuations post-convergence, and Dueling DQN's minimal fluctuations, suggesting stable action predictions. However, as Figure 5b indicates, action stability does not always correlate with reward stability; Dueling DQN maintains a loss magnitude below 10^2, while DQN and Double DQN reach between 10^4 and 10^5. Dueling DQN's lower loss values demonstrate greater stability in action selection and loss magnitude. A more granular analysis of policy entropy over time, depicted in Figure 5c, reveals that while all algorithms exhibit similar upper entropy bounds, their lower bounds differ significantly. Dueling DQN maintains higher entropy, indicating greater action uncertainty and a bias towards exploration. In contrast, Double DQN rapidly approaches a lower entropy bound but fails to achieve convergence, leading to substantial reward oscillations. DQN's steadily decreasing entropy reflects a policy gravitating towards certainty, correlating with superior reward outcomes and demonstrating an effective balance between exploration and exploitation.

Figure 6 compares action convergence profiles between DQN and Double DQN. The upper heatmaps show DQN's actions clustering closely around the optimal angular position, indicating strong align-

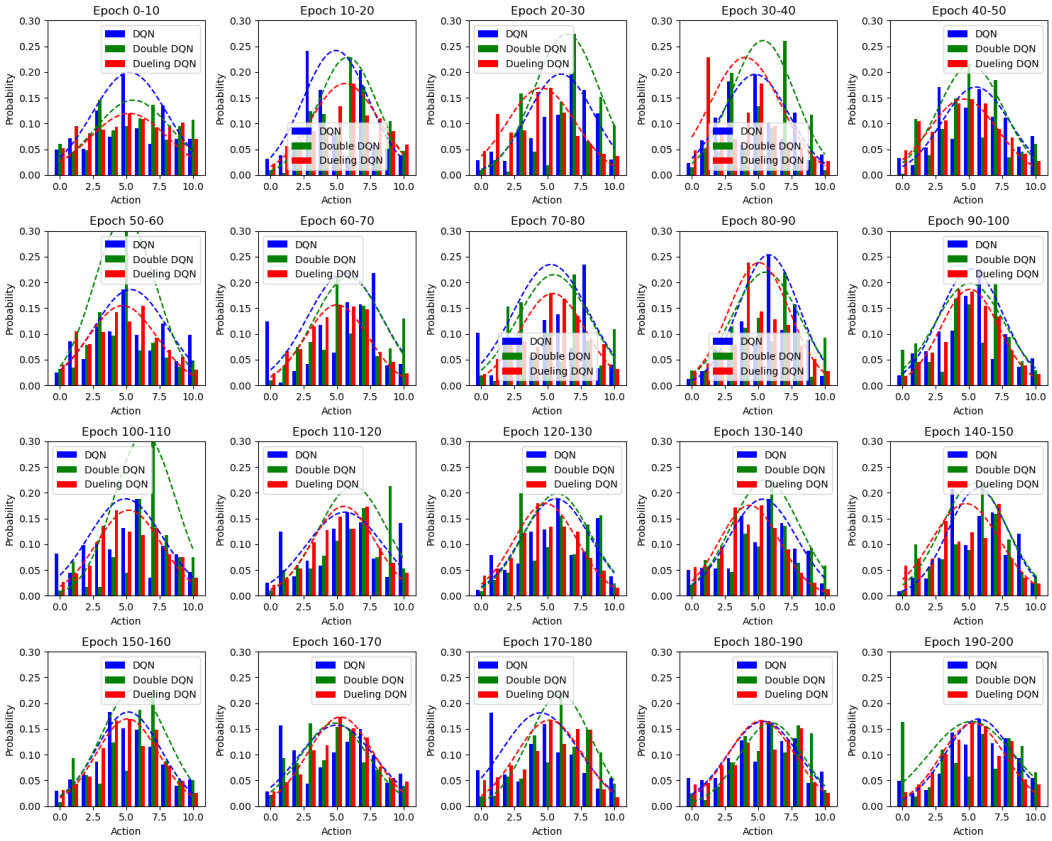


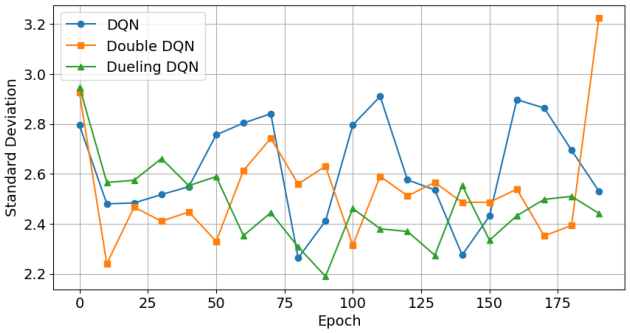
Figure 4: Histogram Analysis of Action Space Exploration in Reinforcement Learning. The figure demonstrates the algorithmic exploration process of DQN, Double DQN, and Dueling DQN, validated by a fitting of normal distributions and quantified by areas outside the expected curves, reflecting each algorithm's approach to exploring the action space.

ment with the APS's desired vertical orientation, while Double DQN displays a more dispersed angular distribution, hinting at less precision. The lower heatmaps illustrate DQN's actions densely concentrated around minimal torque application when the APS is near vertical, whereas Double DQN's actions are more spread out, suggesting a tendency for corrective maneuvers and less stable control. This strategic difference highlights DQN's propensity for fine-tuning actions to maintain the APS's equilibrium, favoring precision and stability, while Double DQN engages in broader exploration, potentially searching for a robust control policy.

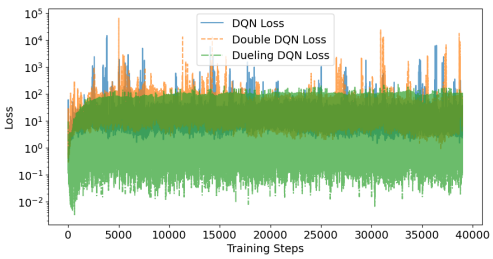
These findings emphasize the need for aligning reinforcement learning algorithm design with environmental complexities to enhance adaptability in real-world conditions. The differential performance of DQN and Dueling DQN across environments highlights the potential for architectural innovations to improve robustness and versatility. Developing algorithms that navigate real-world uncertainties while maintaining efficacy in simulations remains crucial. This analysis underscores the importance of ongoing research to enhance the resilience and adaptability of RL algorithms to the multifaceted nature of real-world environments.

3.3 Error Source Analysis

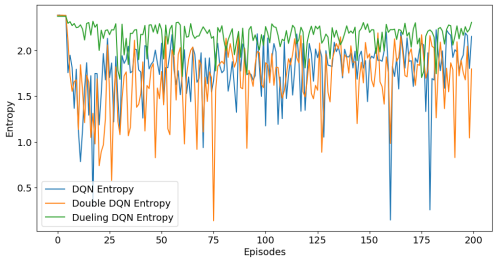
A thorough examination of the discrepancies between estimated and actual measurements revealed critical sources of error. Intrinsic fluctuations in the MPU6050 sensor readings, where a 90-degree value could vary between 85 to 95 degrees, highlight hardware limitations in precision sensing. Additionally, using the action's absolute value instead of direct speed and acceleration measure-



(a) Standard Deviation Trends of Actions Over Training Epochs.



(b) Loss Trends of DQN, Double DQN, and Dueling DQN Throughout Training.



(c) Entropy Variation Over Time for DQN, Double DQN, and Dueling DQN Strategies.

Figure 5: **(a):** presents the fluctuation in action variability for each algorithm, indicating stability and predictability in the decision-making process across epochs. **(b):** This figure delineates the loss magnitude across training steps, with Dueling DQN consistently maintaining loss values below the 10^2 level, exemplifying its efficiency and stability. In contrast, both Double DQN and DQN exhibit loss magnitudes that escalate to above 10^4, indicating higher variability and potential inefficiencies in their learning processes. **(c):** This plot reveals each algorithm's exploration versus exploitation balance, with Double DQN and Dueling DQN reaching similar minimum entropy values around 0.15, indicating a closer approach to optimal policy. In contrast, Dueling DQN maintains a consistently higher entropy, averaging around 2.1, suggesting a strategy that favors exploration and may indicate a more robust but less convergent learning behavior.

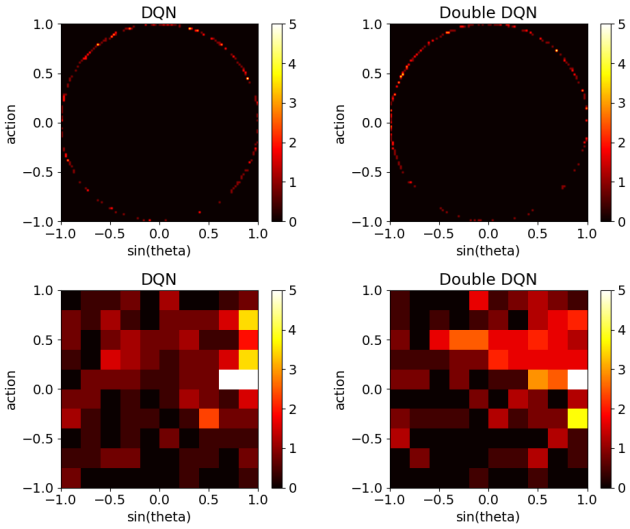


Figure 6: State Visitation Heatmaps: DQN vs. Double DQN. The heatmaps juxtapose the action convergence profiles of DQN and Double DQN, visually depicting their respective policies' precision and stability in maintaining the APS's upright state.

ments in the reward function introduces approximation errors due to the sensor's susceptibility to noise and inaccuracies in rapid measurements. This analysis underscores the complexities of implementing RL algorithms in physical environments and the need for robust sensor fusion techniques and algorithmic adaptations. To foster further research, these observations are documented in our open-source repository for peer replication and scrutiny.

4 Conclusion

This study provides a comprehensive analysis of the applicability and efficacy of various reinforcement learning (RL) algorithms in real-world robotic control, focusing on DQN, Double DQN, and Dueling DQN. Our research highlights the complexities of transitioning from theoretical models to practical applications. DQN demonstrates superior adaptability to real-world environments, often outperforming more complex counterparts. In contrast, Dueling DQN, despite its stability in simulations, shows trade-offs between algorithmic design and environmental fidelity. Our investigation into error sources, such as intrinsic sensor fluctuations and reward function approximations, underscores the need for precision in environmental sensing and robust algorithmic strategies.

Contributions:

APS Environment Development. Created a specialized environment, Angular Positioning Seeker (APS), based on OpenAI Gym for angular positioning tasks, addressing key practical challenges.

Simulation-to-Reality Gap Analysis. Systematically analyzed differences in RL algorithm performance between simulated and physical environments, highlighting real-world factors like hardware noise and dynamic changes.

Physical Experiment Platform. Built a physical experimental platform with Raspberry Pi 4B+ and STM32F103ZET6 microcontrollers for validating RL algorithms in real-world conditions, serving as a practical benchmark.

Extensive Algorithm Implementation. Implemented and validated a range of RL and control algorithms on our physical setup, including REINFORCE, Actor Critic, TRPO, PPO, DDPG, SAC, Behavior Clone, GAIL, PETS, and MBPO. These implementations are available on our GitHub repository, demonstrating the platform's versatility.

Detailed Performance Evaluation. Conducted exhaustive evaluations of multiple algorithms in the APS environment, revealing their strengths and weaknesses in terms of learning efficiency, convergence speed, and stability.

Open Source Resources. Provided all experimental source codes and documentation to ensure reproducibility and community sharing, fostering further research and application of RL algorithms in real-world scenarios.

Future Work

Future research will focus on enhancing the adaptability and robustness of RL algorithms. This includes developing advanced sensor fusion techniques to mitigate hardware inaccuracies and exploring alternative reward function formulations to better capture physical environment dynamics. These efforts aim to bridge the gap between simulated environments and the unpredictable real world, advancing robotics to perform as effectively in real-world applications as in controlled lab settings.

References

[1] P. Kormushev, S. Calinon, and D. G. Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.

[2] M. Eppe, P. D. Nguyen, and S. Wermter. From semantics to execution: Integrating action planning with reinforcement learning for robotic causal problem-solving. *Frontiers in Robotics and AI*, 6:123, 2019.

[3] X. Xiang and S. Foo. Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (pomdp) problems: Part 1—fundamentals and applications in games, robotics and natural language processing. *Machine Learning and Knowledge Extraction*, 3(3):554–581, 2021.

[4] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pages 537–546. PMLR, 2022.

[5] A. Ghadirzadeh, X. Chen, P. Poklukar, C. Finn, M. Björkman, and D. Kragic. Bayesian meta-learning for few-shot policy adaptation across robotic platforms. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1274–1280. IEEE, 2021.

[6] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

[7] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects in 3d with graph networks. *The International Journal of Robotics Research*, 43(4):533–549, 2024.

[8] C. Matl and R. Bajcsy. Deformable elasto-plastic object shaping using an elastic hand and model-based reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3955–3962. IEEE, 2021.

[9] D. Seita, Y. Wang, S. J. Shetty, E. Y. Li, Z. Erickson, and D. Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In *Conference on Robot Learning*, pages 1038–1049. PMLR, 2023.

[10] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[11] V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

[12] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[13] H. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

[14] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[15] A. N. Angelopoulos, S. Bates, et al. Conformal prediction: A gentle introduction. *Foundations and Trends® in Machine Learning*, 16(4):494–591, 2023.

[16] K. Murray and D. Chiang. Correcting length bias in neural machine translation. *arXiv preprint arXiv:1808.10006*, 2018.

[17] S. Kadavath, T. Conerly, A. Askell, T. Henighan, D. Drain, E. Perez, N. Schiefer, Z. Hatfield-Dodds, N. DasSarma, E. Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[18] S. Lin, J. Hilton, and O. Evans. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

[19] H. Ju, R. Juan, R. Gomez, K. Nakamura, and G. Li. Transferring policy of deep reinforcement learning from simulation to reality for robotics. *Nature Machine Intelligence*, 4(12):1077–1087, 2022.

[20] Y. Xiang, X. Wang, S. Hu, B. Zhu, X. Huang, X. Wu, and S. Lyu. Rmbench: Benchmarking deep reinforcement learning for robotic manipulator control. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1207–1214. IEEE, 2023.

[21] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley. Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review. *Robotics*, 10(1):22, 2021.

[22] C. Caponio, P. Stano, R. Carli, I. Olivieri, D. Ragone, A. Sorniotti, and U. Montanaro. Modelling, positioning, and deep reinforcement learning path tracking control of scaled robotic vehicles: Design and experimental validation. *arXiv preprint arXiv:2401.05194*, 2024.

[23] R. Wu, Z. Yao, J. Si, and H. H. Huang. Robotic knee tracking control to mimic the intact human knee profile based on actor-critic reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 9(1):19–30, 2021.

[24] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024.

[25] J. Orr and A. Dutta. Multi-agent deep reinforcement learning for multi-robot applications: A survey. *Sensors*, 23(7):3625, 2023.

[26] D. Han, B. Mulyana, V. Stankovic, and S. Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7):3762, 2023.

[27] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren. Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges. *Robotics*, 8(1):4, 2019.

[28] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[29] X. Yang, Z. Ji, J. Wu, and Y.-K. Lai. Recent advances of deep robotic affordance learning: a reinforcement learning perspective. *IEEE Transactions on Cognitive and Developmental Systems*, 2023.

[30] J. Matas, S. James, and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018.

[31] X. Lin, Y. Wang, J. Olkin, and D. Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[32] B. Balaguer and S. Carpin. Combining imitation and reinforcement learning to fold deformable planar objects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1405–1412. IEEE, 2011.

[33] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[34] J. Yang, D. Sadigh, and C. Finn. Polybot: Training one policy across robots while embracing variability. *arXiv preprint arXiv:2307.03719*, 2023.

[35] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[36] A. Dastider and M. Lin. Damon: Dynamic amorphous obstacle navigation using topological manifold learning and variational autoencoding. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 251–258. IEEE, 2023.

[37] L. X. Shi, Z. Hu, T. Z. Zhao, A. Sharma, K. Pertsch, J. Luo, S. Levine, and C. Finn. Yell at your robot: Improving on-the-fly from language corrections. *arXiv preprint arXiv:2403.12910*, 2024.

[38] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[39] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[40] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

[41] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg. Lazydagger: Reducing context switching in interactive imitation learning. In *2021 IEEE 17th international conference on automation science and engineering (case)*, pages 502–509. IEEE, 2021.

[42] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021.

[43] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *arXiv preprint arXiv:2211.08416*, 2022.

[44] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. *Def*, 2(6):4, 2006.

[45] H. Wang, K. Kedia, J. Ren, R. Abdullah, A. Bhardwaj, A. Chao, K. Y. Chen, N. Chin, P. Dan, X. Fan, et al. Mosaic: A modular system for assistive and interactive cooking. *arXiv preprint arXiv:2402.18796*, 2024.

[46] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[47] E. S. Hu, K. Huang, O. Rybkin, and D. Jayaraman. Know thyself: Transferable visual control policies through robot-awareness. *arXiv preprint arXiv:2107.09047*, 2021.

[48] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik. Robot learning with sensorimotor pre-training. In *Conference on Robot Learning*, pages 683–693. PMLR, 2023.

[49] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.