Reinforcement Learning for Efficient Exploration

1st Hariharan Sureshkumar
Computer Science
Northeastern University
Boston, MA, USA
lnu.harih@northeastern.edu

2nd Ajinkya Bhandare
Electrical and Computer Engineering
Northeastern University
Boston, MA, USA
bhandare.aj@northeastern.edu

3rd Robert Platt
Computer Science
Northeastern University
Boston, MA, USA
rplatt@ccs.neu.edu

***Abstract*—We explore discrete and continuous control methods for efficient robotic exploration in a 2D grid-world environment using Deep Q-Networks (DQN), Actor Critic , Deep Deterministic Policy Gradient (DDPG) and Twin Delayed DDPG (TD3). Our goal is to maximize the coverage of a 100×100 binary grid map generated using Perlin noise, where each cell represents free or occupied space. Unlike discrete action methods such as DQN or A2C, continuous control provides smoother, more expressive navigation strategies, suitable for real-world robotic tasks. By designing a reward function that incentivizes meaningful exploration and penalizes inefficiency, we guide the agent to intelligently traverse the grid. We provide insights into reward shaping, continuous action modeling, and key design decisions that led to improved exploration behavior.**

***Index Terms*—Reinforcement Learning, Discrete Methods, Continuous Control Methods, Exploration**

I. Introduction

Exploration is a critical capability in robotic agents, particularly in environments with sparse rewards or limited observability. Traditional reinforcement learning algorithms, such as DQN, operate in discrete action spaces, which can constrain the flexibility and efficiency of navigation. In tasks that require smooth control or continuous movements such as grid exploration, autonomous driving, or mobile robot mapping, continuous action policies are more appropriate. To address this, we investigate the application of DDPG and TD3, two popular off-policy algorithms designed for continuous control, in the context of 2D grid-world exploration. Our hypothesis is that continuous action outputs will lead to finer control, smoother policy learning, and improved area coverage over time, especially when combined with a well-structured reward scheme. We compare performance and learning trends of both algorithms across multiple metrics and visualize how they adapt to the exploration challenge.

II. Problem Statement

The problem involves training a reinforcement learning agent to explore an unknown 100×100 grid world containing obstacles and free space. The grid is randomly generated using Perlin noise and is binary; each cell is either traversable or blocked. The agent's goal is to cover as much free area as possible in the fewest number of steps while avoiding collisions and unnecessary revisits.

The environment provides only partial observability, with the agent receiving a 5×5 local view centered around its current location. The challenge is compounded by the sparse nature of the rewards, the potential for repetitive movement, and the long time horizons required to explore large maps. This motivates the use of continuous action spaces for precision navigation and reward shaping to ensure efficient learning in such a sparse-reward setting.

The challenge lies in training a reinforcement learning agent to effectively explore a large, unknown 100x100 grid world. This environment is characterized by a binary distribution of obstacles and traversable areas, generated using Perlin noise to simulate realistic, complex terrain. The agent's objective is to maximize its coverage of the free space within the grid, doing so in a minimal number of steps, while avoiding collisions and redundant exploration.

Several factors complicate this task. The agent possesses only partial observability, limited to a 3x3 local view centered on its current position. Additionally, rewards are sparse, and the potential for the agent to fall into repetitive movement patterns is high. The large size of the grid also leads to long time horizons, further increasing the difficulty of efficient exploration. These challenges suggest the need for a continuous action space to enable precise navigation, as well as the implementation of reward shaping techniques to guide learning in this sparse reward setting.

To address these challenges, the agent's state representation is augmented. It combines the agent's historical knowledge of visited locations with the new information gained from its current partial observation. Specifically, the state is represented as a 100x100 grid. In this grid, traversable areas are marked as 0 or 1, the agent's past trajectory is represented by 0.5, and the agent's current location is also tracked. The grid environment is generated using Perlin noise, followed by a flood fill operation from the origin to ensure that all reachable free space is connected. Unreachable free space is then filled in as obstacles, and the environment is designed to have more than 60

III. Proposed Solution

A. Continious Algorithms

We propose using two continuous control algorithms: DDPG and TD3, in addition to 2 discrete algorithms DQN and Actor Critic to solve this grid exploration problem. Both

of the continious algorithms operate over a continuous action space

$$[-1, 1]^2$$

Where the actor network outputs a movement vector (dx,dy). This vector is scaled and rounded to translate to grid movement while allowing more nuanced transitions than traditional discrete directions. To promote efficient learning and prevent the agent from falling into suboptimal local behaviors, we carefully designed a reward function.

- The agent receives +5 for exploring a new free cell
- -0.1 for revisiting an already seen cell
- -2 for hitting an obstacle
- -1 for remaining in place

Additionally, to globally encourage coverage, we add a bonus of +10 x coverage x ratio at each step. TD3 improves on DDPG with techniques such as twin critics to reduce overestimation bias, delayed policy updates, and target policy smoothing with clipped noise.

B. Discrete Algorithms

We use two discrete deep reinforcement learning algorithms DQN and Actor Critic to solve this exploration. DQN is a value-based method that approximates the Q-value function using a neural network, where the Q-value represents the expected return of taking an action in a given state and following the optimal policy thereafter. It uses experience replay buffer to break correlations in the training data and a target network to stabilize learning by decoupling the target Q-values from the rapidly changing estimates.

Actor-Critic methods combine both value-based and policy-based learning. The ”actor” learns a policy function that maps states to actions, while the ”critic” evaluates the quality of actions taken by estimating the value function. This dual structure allows Actor-Critic algorithms to benefit from the stability of value-based methods and the direct policy optimization of policy-gradient methods. Initially we started with a simpler reward structure with large penalty for colliding and small constant penalty for every time step. This prevented the agent from learning because the best action according to the agent was to collide and end the episode early as the number of possible cells till end state can be very large ($>10^5$).

For these 2 algorithms, we found the following reward structure to perform better than the simpler one. We give a reward of:

- +10 for discovering a new cell
- -1 for revisiting an old cell
- -100 for colliding with a wall

This structure helps us prioritize exploration of new cells with positive reward, penalize revisiting older cells, as well as colliding with walls.

IV. Implementation and Testing

A. DDPG

The Deep Deterministic Policy Gradient (DDPG) agent was implemented with a single actor-critic architecture operating in a continuous action space. The actor network outputs a two-dimensional movement vector (dx,dy) scaled between [-1,1], which is clipped and applied to navigate the agent within the grid. The critic network estimates the Q-value of each state-action pair, guiding the actor during policy updates. Target networks for both actor and critic were used, and their weights were softly updated over time to improve stability during training.

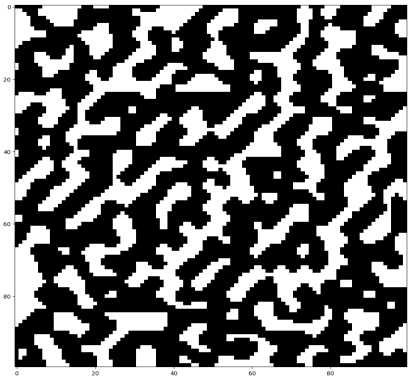

Fig. 1. Example of a generated grid

DDPG training was conducted across 500 to 2000 episodes, with each episode consisting of up to 10,000 steps. To encourage exploration, we added Gaussian noise to the actions, with the noise scale decaying as training progressed. The agent's replay buffer stored transitions to enable off-policy learning via minibatch sampling. Although DDPG showed signs of learning, its single-critic design made it more susceptible to overestimating Q-values, resulting in high variance and slower convergence in some cases. Nevertheless, with reward shaping and normalization, it demonstrated consistent improvements in area coverage.

B. Twin Delayed DDPG (TD3)

Twin Delayed DDPG (TD3) was built on the DDPG framework but introduced critical modifications to enhance performance and stability. The TD3 agent used two critic networks to compute separate Q-value estimates, and during training, the minimum of the two was used to reduce overestimation bias. Actor updates were delayed relative to critic updates, ensuring the policy was optimized based on more accurate value estimates. Additionally, noise was added to the target policy during critic target computation, with clipping applied to prevent excessive deviation.

TD3 was trained under the same environment and hyperparameter constraints as DDPG, with 500 to 2000 episodes and 10,000 steps per episode. We normalized state inputs, applied decaying exploration noise, and used a shared replay buffer for off-policy learning. TD3 consistently outperformed DDPG in both reward and coverage metrics, showing smoother learning curves and higher overall stability. The algorithm's twin critics and delayed policy updates made it more robust to noise and reward variance, allowing it to better adapt to the structured reward signals and complex grid layout.

C. Deep Q Learning (DQN)

The DQN is a value-based method that approximates the Q-value function using a neural network, where the Q-value represents the expected return of taking an action in a given state and following the optimal policy thereafter. We update the Q network every 4 steps while we update the target network every 1,000 steps. This helps us stabilize the training process with rapidly changing Q estimates. In addition, it uses an experience replay buffer to break the correlations in the training data.

The DQN agent involves 3 convolution layers followed by 3 fully connected layers to predict the Q value for each state. We run the training 100,000 steps with an epsilon greedy action selection for exploration. We follow an exponential epsilon decay schedule to reduce exploration over time and take greedy learned actions.

D. Actor Critic Method

Actor-Critic method employs two similar networks with one estimating the value of every state and the other is a policy network taking actions for a state. We start by generating and episode by taking actions from the action policy. We then compute the Advantage function for each state computing the loss for actor using log and the loss for value function by mean squared error of actual reward obtained from each state and estimated expected reward for each state.

For running the algorithm, we use the Adam optimizer for both the networks and with a learning rate of $5x10^{-}3$. The models for actor and critic have 3 convolution layers followed by 3 fully connected layers leading to a 4 dimensional and 1 dimensional output. The network was trained for 50,000 which saw a decreasing loss, but a stagnant performance.

V. Results

Across training runs, both DDPG and TD3 showed increasing reward and coverage trends, with TD3 demonstrating faster convergence and lower variance in performance. In early episodes, the agents learned to avoid obstacles and seek new areas using the reward shaping, while later episodes showed improved trajectory planning and fewer redundant movements. TD3 outperformed DDPG in terms of both smoothed total reward and grid coverage percentage, achieving over 80% coverage in the best cases. Reward plots showed clearer trends with TD3, reflecting the algorithm's robustness. The decaying noise policy and reward shaping together helped overcome the sparse reward landscape of the environment.

A. Deep Deterministic Policy Gradient (DDPG) results

Figure 2 shows the performance of the DDPG agent trained for 200 episodes without our updated reward function. The reward curve is highly unstable and mostly stays at large negative values. This behavior reflects that the agent receives very little useful feedback during training largely due to the sparse rewards. The agent often fails to make progress and repeatedly incurs penalties for collisions or idle actions, leading to erratic spikes and no consistent learning.

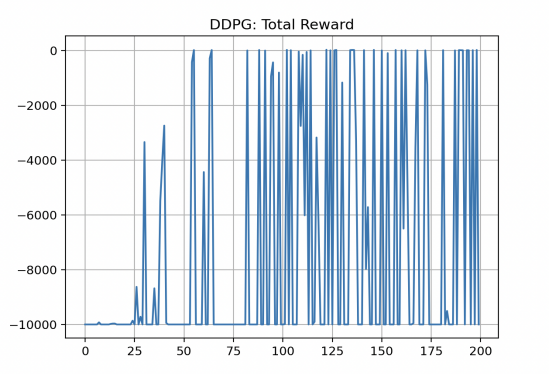

Fig. 2. DDPG algorithm without our updated reward function and trained on 200 episodes

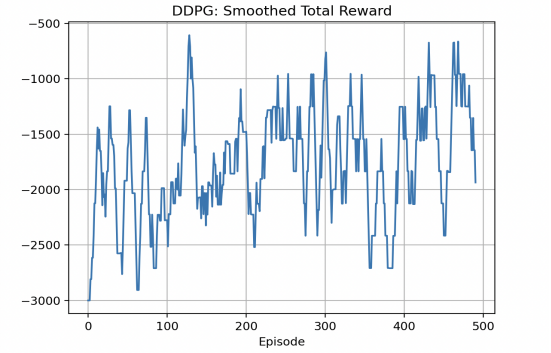

Fig. 3. DDPG algorithm learning with our updated reward function and trained on 500 episodes

Figure 3 presents the same DDPG agent trained for 500 episodes with our custom reward shaping applied. The reward plot is noticeably smoother and shows a general upward trend. This improvement indicates that the agent is learning to associate actions with meaningful exploration outcomes. The shaped rewards — such as penalties for revisiting cells and bonuses for new cell exploration — help guide the policy updates in a more structured way, resulting in improved behavior over time.

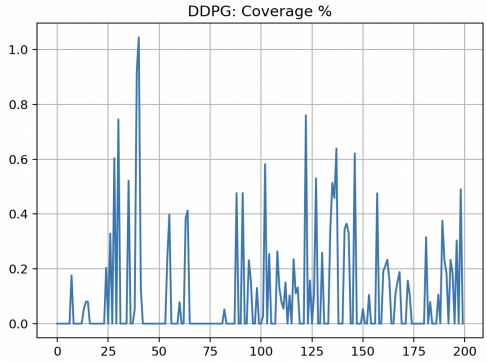

Fig. 4. DDPG algorithm exploration coverage without our updated reward function and trained on 200 episodes

Figure 4 displays the exploration coverage of the agent without reward shaping (from the same run as Fig. 2). The coverage remains very limited, with only a few episodes achieving notable exploration. This aligns with the agent's inability to learn useful behavior under sparse rewards — most episodes result in random movement, looping behavior, or early termination due to collisions.

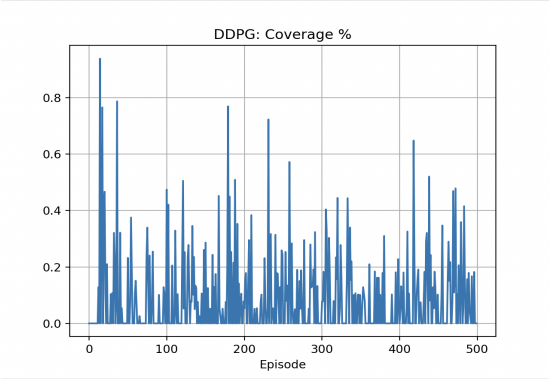

Fig. 5. DDPG algorithm exploration coverage with our updated reward function and trained on 500 episodes

Figure 5, on the other hand, shows the coverage results for the agent trained using reward shaping (same run as Fig. 3). The spikes in this plot are more frequent and significantly higher, indicating that the agent learns to explore larger portions of the grid. The coverage pattern supports the effectiveness of the shaped reward terms, especially the incentives for discovering new free cells and discouraging idle or repeated movements.

B. Twin Delayed DDPG (TD3) results

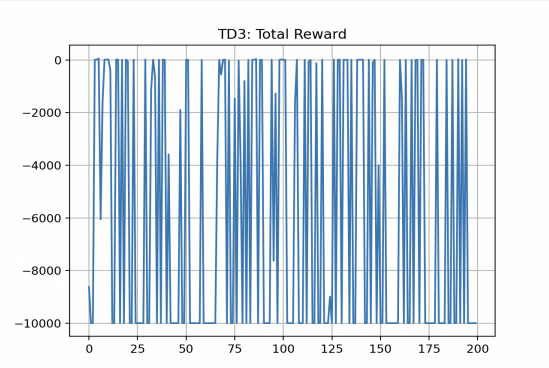

Fig. 6. TD3 algorithm without our updated reward function and trained on 200 episodes

Figure 6 shows the raw total reward of the TD3 agent trained for 200 episodes without our updated reward function. The rewards are highly volatile and mostly stuck around large negative values, with little indication of policy improvement. This reflects the difficulty of learning in a sparse reward setting, where agents receive little feedback on what actions lead to better exploration or goal-directed behavior.

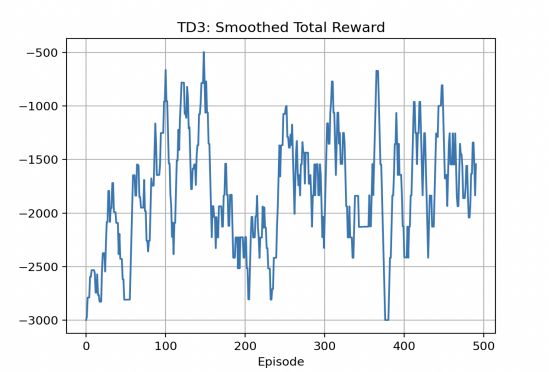

Fig. 7. TD3 algorithm learning with our updated reward function and trained on 500 episodes

In Figure 7 we illustrate the performance of TD3 after applying our updated reward shaping and extending training to 500 episodes. The smoothed reward curve reveals a much more consistent and improving trend compared to Fig. 6. This shows that the TD3 agent benefits significantly from structured reward feedback, leveraging its twin-critic setup and delayed actor updates to learn a more stable and effective policy.

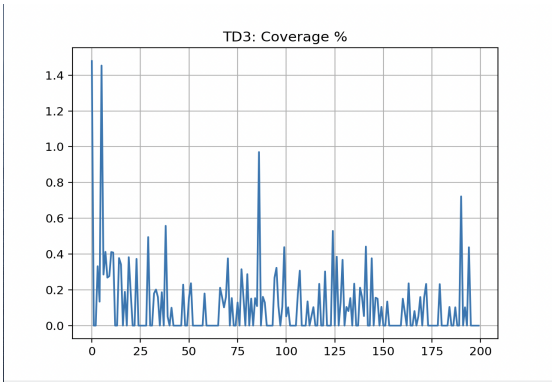

Fig. 8. TD3 algorithm exploration coverage without our updated reward function and trained on 200 episodes

Figure 8 presents the exploration coverage of TD3 without reward shaping. Like the earlier DDPG result, the coverage is minimal and inconsistent. The agent fails to generalize exploration behavior, and most episodes result in poor map traversal and little gain in environment knowledge.

In Figure 9 we show how the updated reward terms dramatically improve coverage. The TD3 agent is able to cover more of the environment in many episodes, with multiple spikes approaching 1.0 (i.e., near-complete coverage). This supports that TD3, with its robust architecture and shaped rewards, is better suited to handle the challenges of sparse feedback in continuous grid-world exploration.

C. DQN results

The Figure 10 focuses on reward progression for the DQN agent. Unlike A2C, the DQN rewards remain fairly steady over the course of training, but mostly within a narrow and

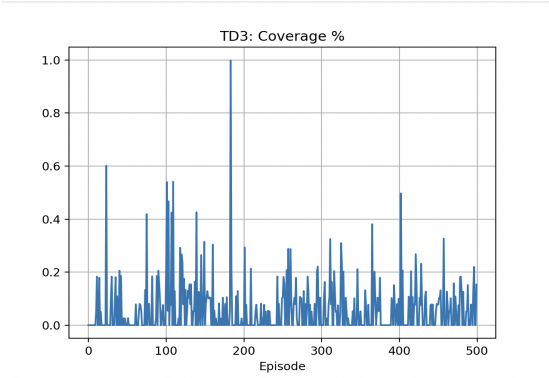

Fig. 9. TD3 algorithm exploration coverage with our updated reward function and trained on 500 episodes

consistently negative band. While this shows lower volatility compared to A2C, it also implies that the DQN agent failed to discover better policies and got stuck in a local minimum early in training.

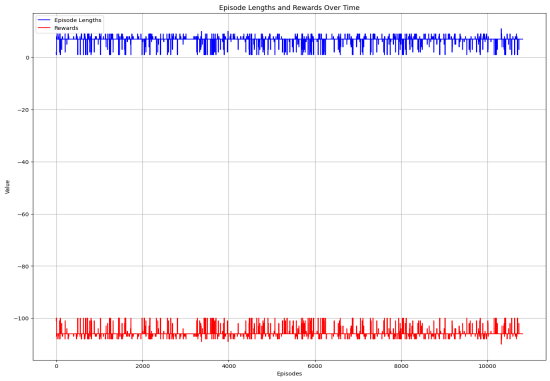

Fig. 10. DQN algorithm reward per episode and the length each episode

The loss for the network over the period of training as shown in Figure 11 is decreasing gradually for the first half of the training, but settles down to a constant value later. The loss demonstrates high variace which is similar to the variance in reward and episode length as observed for this method.

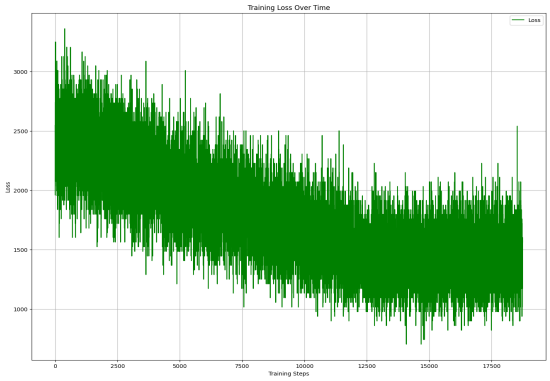

Fig. 11. The loss for the DQN network as we train through the steps

D. Actor Critic results

In the Figure 12, we observe the evolution of episode lengths and rewards over time. The reward curve (red) shows a highly negative trend initially, with significant volatility. While some early episodes achieve very high positive rewards, they quickly give way to consistently negative rewards as training progresses. This may indicate that the policy struggled to generalize or adapt effectively to the environment, leading to suboptimal performance.

The episode lengths (blue), while showing some variability, remain relatively low and do not show a clear increasing trend. This, combined with the flat and negative reward curve, implies that the agent did not consistently improve or explore deeper into the environment.

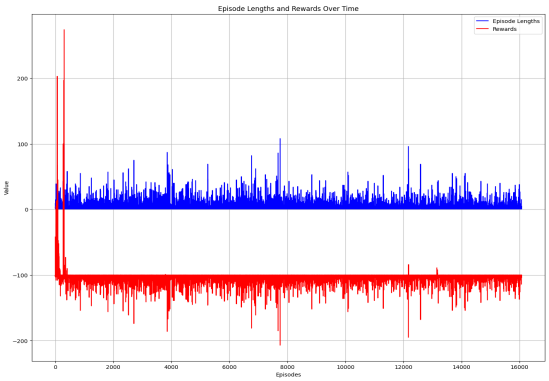

Fig. 12. Actor Critic algorithm reward per episode and the length of each episode

As shown in the Figure 13, the training loss for the A2C agent initially exhibits high variance and large spikes during the early training steps. This is common in reinforcement learning due to unstable updates when the policy and value functions are still being initialized. However, the loss gradually stabilizes as training progresses, converging to lower values with reduced variance. This suggests that the policy and value networks learned more consistent updates over time, reflecting improved training stability.

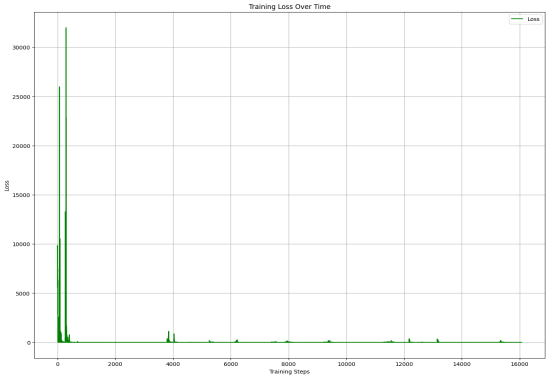

Fig. 13. The loss for the Actor Critic network as we train through the steps

VI. Conclusion and Future work

Across training runs, both DDPG and TD3 showed increasing reward and coverage trends, with TD3 demonstrating faster convergence and lower variance in performance. In early episodes, the agents learned to avoid obstacles and seek new areas using the reward shaping, while later episodes showed improved trajectory planning and fewer redundant movements.

TD3 outperformed DDPG in terms of both smoothed total reward and grid coverage percentage, achieving over 80% coverage in the best cases. Reward plots showed clearer trends with TD3, reflecting the algorithm's robustness. The decaying noise policy and reward shaping together helped overcome the sparse reward landscape of the environment.

For the DQN and Actor Critic, the agents increased coverage and moved a few steps. The actor critic agent learned faster compared to the DQN network but remained stagnant after a while with occasional spikes in between. The loss quickly goes down for Actor critic method while it decreases gradually for the DQN network and later settles to a low value not becoming zero. The DQN method learns to move for a few steps but then gets stuck where it takes random steps leading to spikes but has a baseline episode length which does not increase. We assume this problem with DQN is because it involves estimating accurate Q value for a complex state space without any episode reaching to the end. Hence, the policy based algorithms like Actor critic, DDPG and TD3 perform better.

References

[1] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al., "Continuous control with deep reinforcement learning," arXiv:1509.02971, 2016.

[4] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. Machine Learning (ICML)*, 2018.

[5] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. 34th Int. Conf. Machine Learning (ICML)*, 2017.

[6] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. 16th Int. Conf. Machine Learning (ICML)*, 1999.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[8] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Proc. Robotics: Science and Systems (RSS)*, 2005.

[9] M. Young, *The Technical Writer's Handbook*. Mill Valley, CA: University Science, 1989.

[10] D. Silver, A. Huang, C. J. Maddison, et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[12] Y. Burda, H. Edwards, D. Pathak, et al., "Large-scale study of curiosity-driven learning," arXiv:1808.04355, 2018.

[13] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.

[14] M. Andrychowicz, F. Wolski, A. Ray, et al., "Hindsight experience replay," in *Proc. 31st Conf. Neural Information Processing Systems (NeurIPS)*, 2017.