Deep Reinforcement Learning for Autonomous Ground Vehicle Exploration Without A-Priori Maps

Shathushan Sivashangaran   shathushansiva@vt.edu
Department of Mechanical Engineering
Virginia Tech
Blacksburg, VA 24061, USA

Azim Eskandarian   eskandarian@vt.edu
Department of Mechanical Engineering
Virginia Tech
Blacksburg, VA 24061, USA

Corresponding Author: Shathushan Sivashangaran

Abstract

Autonomous Ground Vehicles (AGVs) are essential tools for a wide range of applications stemming from their ability to operate in hazardous environments with minimal human operator input. Effective motion planning is paramount for successful operation of AGVs. Conventional motion planning algorithms are dependent on prior knowledge of environment characteristics and offer limited utility in information poor, dynamically altering environments such as areas where emergency hazards like fire and earthquake occur, and unexplored subterranean environments such as tunnels and lava tubes on Mars. We propose a Deep Reinforcement Learning (DRL) framework for intelligent AGV exploration without a-priori maps utilizing Actor-Critic DRL algorithms to learn policies in continuous and high-dimensional action spaces directly from raw sensor data. The DRL architecture comprises feedforward neural networks for the critic and actor representations in which the actor network strategizes linear and angular velocity control actions given current state inputs, that are evaluated by the critic network which learns and estimates Q-values to maximize an accumulated reward. Three off-policy DRL algorithms, DDPG, TD3 and SAC, are trained and compared in two environments of varying complexity, and further evaluated in a third with no prior training or knowledge of map characteristics. The agent is shown to learn optimal policies at the end of each training period to chart quick, collision-free exploration trajectories, and is extensible, capable of adapting to an unknown environment without changes to network architecture or hyperparameters. The best algorithm is further evaluated in a realistic 3D environment.

Keywords: Autonomous Ground Vehicles, Cognitive Navigation, Deep Reinforcement Learning

1. INTRODUCTION

Autonomous Ground Vehicles (AGVs) are indispensable tools for mapping uncharted terrain, Search and Rescue (SAR) missions, disaster response, military operations, mining, and extraterrestrial planetary exploration owing to their ability to operate in hazardous, unstructured environments reliably with minimal input from a human operator [1, 2]. Perceiving the environment, and planning trajectories are key components of AGV navigation.

Mobile robot trajectories require optimization for shortest path, minimum energy consumption and training time [2]. Conventional AGV navigation algorithms are dependent on specific environmental configurations [3] which limits their effectiveness in adapting to information poor, dynamically changing environments such as areas where emergency hazards like fire and earthquake occur, and unexplored subterranean environments such as tunnels, caves and lava tubes on Mars.

Conventional navigation algorithms comprise graph search algorithms such as Dijkstra, A* and D* [4] that are well-defined and simple to use but are inefficient in complex, dynamic environments and have poor robustness to noise interference and errors in the environment model, random sampling algorithms such as Probability Graph Method (PGM) and Rapid exploration Random Tree (RRT) [5] that select random scatter points in the entire environment space to search for the optimal path between the starting and end points making them susceptible to poor real-time performance, sub-optimal solutions and high computation cost, Artificial Potential Field (APF) [6] that is efficient but prone to local minima traps, and nature inspired algorithms such as fuzzy logic that is robust, but requires prior knowledge in the form of user defined knowledge based logic and rules, and Genetic Algorithm (GA) [7] which is ideal for the global optimal solution and suitable for complex problems, but has poor local search ability and slow convergence rate.

Recent advancements in Artificial Intelligence (AI), sensors, communication and computer technology facilitate intelligent AGVs capable of high autonomy. Motion planning models that incorporate Artificial Neural Networks (ANNs) and Actor-Critic Reinforcement Learning (RL) enable robotic systems to learn optimal, end-to-end policies in continuous and high-dimensional action spaces directly from characteristics of high-dimensional sensory input data to intelligently select goal driven actions in obstacle filled unstructured terrain in the absence of prior knowledge and detailed maps [8–11].

Numerous works have focused on utilizing Deep Reinforcement Learning (DRL) for task-driven navigation aided by GPS observations to specific goal positions generated by a high-level planner. In [12], the authors train an indoor robot equipped with camera, GPS and compass sensors in simulation, to navigate to a target location from a random initial position in an unseen map, for over 180 days of GPU-time parallelly with 64 GPUs, in 3 days using Decentralized Distributed Proximal Policy Optimization (DD-PPO) to achieve exceptional post-training performance.

However, task-independent exploration of a new environment to facilitate various down-stream applications has received significantly less attention [13]. Effective exploration in GPS-denied environments such as indoor and subterranean environments encountered in SAR, military and extraterrestrial planetary exploration require even greater training times, and different DRL approaches.

Simultaneous Localization And Mapping (SLAM) enables AGVs to simultaneously estimate vehicle state utilizing on-board sensors and construct a model of the environment the sensors perceive [14]. The inclusion of LIDAR and visual SLAM in the perception pipeline is a key enabler for contemporary AGV navigation in environments that are GPS-denied with no access to a-priori maps [15]. In [16], the authors propose Active Neural SLAM (ANS), a modular navigation model that integrates SLAM with DRL to effectively explore an unknown environment. ANS comprises a neural SLAM module to predict the map of the surrounding environment and the agent pose, a global policy that uses a convolutional neural network to output a long-term goal using the spatial map, agent pose and visited locations as inputs trained using Proximal Policy Optimization (PPO) [17] to maximize coverage, and a local policy that utilizes RGB observations and a short-term goal derived from the long-term goal as inputs to a recurrent neural network to obtain the navigational action.

SLAM is subject to mapping and pose estimate errors as a consequence of estimating sequential motion, and a modular navigation model is computationally demanding as it comprises multiple ANNs for each subtask. End-to-end navigation utilizing a single DRL network achieved with reward shaping without a dependence on mapping and long-term goal generation is of benefit to early stage task-independent exploration of unknown environments in adverse environmental conditions for data collection to build more accurate maps, and facilitate down-stream applications.

Moreover, on-policy Actor-Critic Deep Reinforcement Learning (DRL) algorithms such as PPO, that are most commonly used in prior work, are robust to hyperparameter tuning and straightforward to implement, but are sample inefficient as these require new training samples for every policy update, which makes learning an effective policy for complex tasks computationally exorbitant. Off-policy Actor-Critic DRL algorithms such as Deep Deterministic Policy Gradient (DDPG) [18], Twin Delayed Deep Deterministic Policy Gradient (TD3) [19] and Soft Actor-Critic (SAC) [20] reuse past experience stored in a replay buffer for learning, thus have higher sample efficiency.

Given the potential of DRL for AGV navigation in information poor environments, this paper presents and evaluates a DRL framework with shaped rewards for cognitive AGV navigation solely utilizing distance measurement observations obtained using LiDAR or camera RGB-D point cloud, and compares state-of-the-art off-policy DRL algorithms' ability to safely navigate and explore obstacle filled terrain without prior knowledge of environment characteristics. Range measurements were chosen as observations over RGB images as they are more efficient, since they provide a complete understanding of the surrounding environment at a fraction of the data size. Animals such as bats use echolocation for navigation [21] solely utilizing an understanding of distances to objects, hence it is reasonable to expect a DRL agent to learn a reliable, robust navigation policy exclusively utilizing range measurements. The decrease in the size of the observation space enables faster learning of policies in more complex environments that require a large number of training samples.

The rest of the paper is organized as follows: Section II provides a background on DRL. The proposed DRL framework is detailed in Section III. The training and evaluation methodologies are described in Section IV. The results are presented and analyzed in Section V, and finally Section VI provides closing remarks and future work directions.

2. BACKGROUND ON DEEP REINFORCEMENT LEARNING

RL is a Machine Learning (ML) framework inspired by trial-and-error animal learning to train agents that interact with the surrounding environment by promoting or discouraging actions utilizing reward feedback signals designed to gauge effectiveness of executed actions. Deep Learning (DL), a key ML component, utilizes ANNs to form an abstract, distinguishable high-level representation from low-level input features. DRL algorithms combine DL and RL to extract unknown environment features from high-dimensional input data utilizing ANNs, and decide control actions using RL. FIGURE 1, portrays the DRL framework.
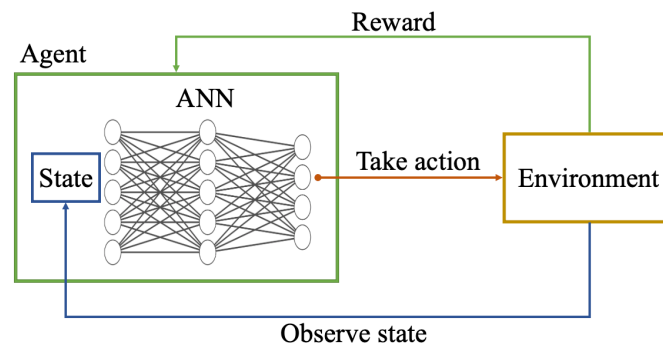

Figure 1: Schematic of deep reinforcement learning framework.

A RL agent observes its environment s_i at each time step t, and selects an action a_i from action space A, conforming to a learned policy $\pi(a_i \mid s_i)$ that maps states to actions. The expectation of a discounted, accumulated reward $R_i = \Sigma_{k=0}^{\infty} \gamma^k rw_{i+k}$ at each state is maximized during learning, where $\gamma \in (0,1]$ is the discount factor, and rw_i is the scalar reward signal for selecting action a_i [22].

2.1 Actor-Critic Framework

An actor-critic framework utilizing deep function approximators that combines both value-based and policy-based RL is the preferred method to learn policies in continuous and high-dimensional action spaces, required for robotics applications. This method leverages the joint computing and decision-making abilities of the actor and critic neural networks to yield low variance and fast speeds when updating gradients. FIGURE 2, illustrates the Actor-Critic framework.

The actor network strategizes an action output selected from a continuous action space using policy gradient, utilizing the current state as the input. The critic evaluates the chosen actions and outputs the associated approximate Q-value for the current state and selected action using an approximated value function to counter the large variance in the policy gradients. In off-policy algorithms, sample data accumulated in a replay buffer is utilized to update and approximate the value function yielding higher sample efficiency than on-policy algorithms. The two networks compute the action prediction for the current state at each time step to generate a temporal-difference error signal.
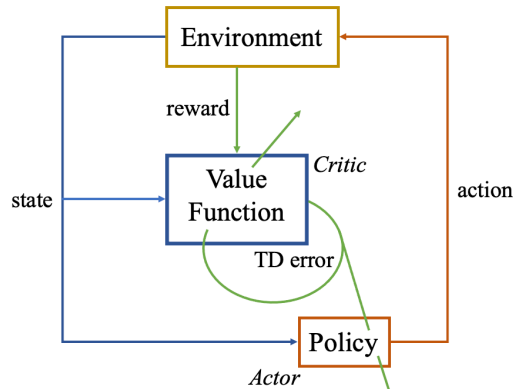
Figure 2: Schematic of Actor-Critic framework.

2.2 Deep Deterministic Policy Gradient

DDPG [18] is a model-free, off-policy actor-critic RL algorithm that combines ANNs with the actor-critic representation of standard Deterministic Policy Gradient (DPG) [23] to successfully implement control sequences in a continuous action space. The actor, $\pi(s \mid \theta)$ and critic, $Q(s, a \mid \phi)$ each comprise fully-linked, two-layer feedforward ANNs with Rectified Linear Unit (ReLU) activation functions.

The loss L is minimized across all sampled experiences to update the critic parameters, ϕ,

$$L = \frac{1}{M} \sum_{i=1}^{M} (y_i - Q(s_i, a_i \mid \phi))^2 \tag{1}$$

Here M is a random mini-batch of experiences, and y_i is the target value function computed as follows,

$$y_i = R_i + \gamma Q_t(s_{i+1}, \pi_t(s_{i+1} \mid \theta_t) \mid \phi_t) \tag{2}$$

θ_t and ϕ_t are parameters of the target actor π_t and target critic Q_t respectively, that have the same structure and parameterization as π and Q. The agent periodically updates θ_t and ϕ_t using the latest θ and ϕ values to improve the stability of the optimization.

The actor parameters, θ are updated using a sampled policy gradient $\nabla_\theta J$ to maximize the expected discounted reward,

$$\nabla_\theta J \approx \frac{1}{M} \sum_{i=1}^{M} G_{ai} G_{\pi i} \tag{3}$$

Here G_{ai} is the gradient of the critic output with respect to the action selected by the actor network computed as follows,

$$G_{ai} = \nabla_a Q(s_i, \pi(s_i \mid \theta) \mid \phi) \tag{4}$$

$G_{\pi i}$ is the gradient of the actor output with respect to its parameters,

$$G_{\pi i} = \nabla_\theta \pi(s_i \mid \theta) \tag{5}$$

2.3 Twin-Delayed Deep Deterministic Policy Gradient

TD3 is designed to improve learned policies by preventing overestimation of the value function [19]. Two Q-value functions are learned simultaneously, and the minimum is used for policy updates. Moreover, the policy is updated less frequently than the Q-value function to further improve learned policies.

The parameters of the critic, $Q_k(s, a \mid \phi_k)$, where $k = 2$ is the number of critics, are updated by minimizing the loss L_k as follows,

$$L_k = \frac{1}{M} \sum_{i=1}^{M} (y_i - Q_k(s_i, a_i \mid \phi_k))^2 \tag{6}$$

The target value function y_i is computed as follows,

$$y_i = R_i + \gamma \min_k (Q_{tk}(s_{i+1}, clip(\pi_t(s_{i+1} \mid \theta_t) + \varepsilon) \mid \phi_{tk})) \tag{7}$$

Here θ_t and ϕ_{tk} are parameters of the target actor π_t and target critics Q_{tk}, and ε is noise added to the computed action to promote exploration. The action is clipped based on the noise limits.

The actor parameters are updated similar to DDPG using Equation (3) where G_{ai} is computed as follows and $G_{\pi i}$ is computed as in Equation (5).

$$G_{ai} = \nabla_a \min_k (Q_k(s_i, \pi(s_i \mid \theta) \mid \phi)) \tag{8}$$

2.4 Soft Actor-Critic

SAC [20], similar to DDPG and TD3, is a model-free, off-policy actor-critic RL algorithm. In addition to maximizing the long-term expected reward, SAC maximizes the entropy of the policy, which is a measure of the policy uncertainty at a given state. A higher policy entropy promotes exploration, hence the learned policy balances exploitation and exploration of the environment.

The agent utilizes a stochastic actor that outputs mean and standard deviation, using which an unbounded action is randomly selected from a Gaussian distribution. The entropy of the policy is computed during training for the given observation using this unbounded probability distribution. Bounded actions that comply with the action space are generated by applying *tanh* and scaling operations to the unbounded action.

The critic parameters are updated at specific time step periods by minimizing the loss function in Equation (6), similar to TD3 for k critics.

The target value function y_i is computed as the sum of the minimum discounted future reward from the critic networks R_i, and the weighted entropy as follows,

$$y_i = R_i + \gamma \min_k (Q_{tk}(s_{i+1}, \pi(s_{i+1} \mid \theta) \mid \phi_{tk}))$$

$$-\alpha ln\pi(s_{i+1} \mid \theta) \tag{9}$$

Here α is the entropy loss weight. The entropy weight is updated by minimizing the loss function, L_α where H is the target entropy as follows,

$$L_\alpha = \frac{1}{M} \sum_{i=1}^{M} (-\alpha ln\pi(s_i \mid \theta) - \alpha H) \tag{10}$$

The stochastic actor parameters are updated by minimizing the objective function J_π,

$$J_\pi = \frac{1}{M} \sum_{i=1}^{M} (-\min_k (Q_{tk}(s_i, \pi(s_i \mid \theta) \mid \phi_{tk})) + \alpha ln\pi(s_i \mid \theta)) \tag{11}$$

3. PROPOSED DEEP REINFORCEMENT LEARNING FRAMEWORK

This section presents the DRL framework, and reward design for quick, efficient and collision-free AGV exploration in new, unknown environments.

3.1 Network Architecture

In order to maximize the long-term reward, designed to encourage quick, efficient, and collision-free exploration of the environment, the DRL agent makes strategic linear and angular velocity action decisions for the current time step, v_t and ω_t. These decisions are based on LiDAR or camera RGB-D point cloud range measurements from the current and previous time steps r_t and r_{t-1}, the previous time step's action $a_{t-1} = (v_{t-1}, \omega_{t-1})$, and the corresponding reward value, R. The proposed DRL architecture for AGV exploration is shown in FIGURE 3.
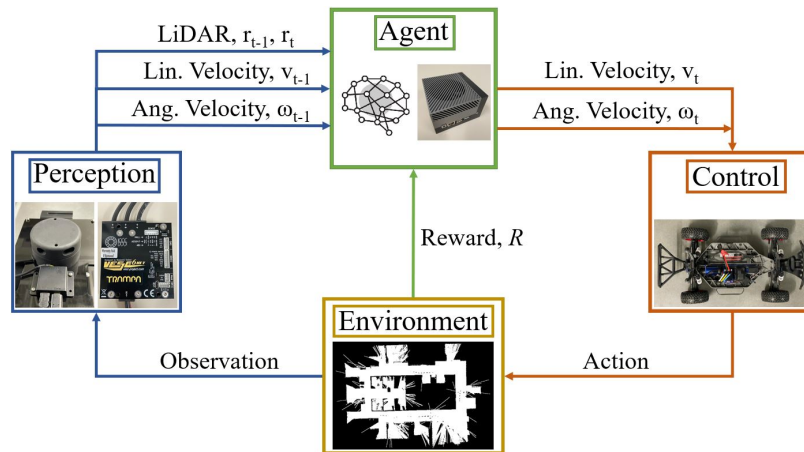
Figure 3: Deep Reinforcement Learning framework for Autonomous Ground Vehicle exploration without maps.

The agent updates the policy at each time step during training using the selected action, the prior and current observations after executing the action, and scalar reward feedback, with the objective of maximizing the long-term reward to promote collision-free exploration.

The addition of global position information obtained via GPS to the observation space will improve learning sample efficiency, and can encourage more efficient exploration, however it is not viable in environments that are underground, or inside buildings. To mitigate this, the reward is specifically shaped to encourage the agent to navigate previously unexplored regions solely utilizing range and odometry measurements.

3.2 Reward Shaping

The reward function is application specific, and designed to encourage the agent to explore its environment efficiently, quickly and safely without collisions. To compare the exploration capabilities of DDPG, TD3 and SAC, a minimalist reward function is shaped as presented in [9],

$$R = 0.0075r^2 + 1.5v^2 - 0.6\omega^2 \tag{12}$$

A positive reward is applied to the square of the minimum range measurement, r to incentivize obstacle avoidance. This reward is highest when the agent is at a greater distance from obstacles, encouraging the generation of paths devoid of obstacles. The agent is additionally rewarded for swift navigation through positive reinforcement of linear velocity, v. To encourage efficient exploration, a negative reward is applied to angular velocity, ω to discourage repeated circular motion in the same vicinity. High coefficients for r^2 and v^2 lead to a compromise between obstacle avoidance ability and exploratory behavior, hence a balance was determined through trial-and-error experimentation to prioritize both exploration, and collision avoidance.

The presented DRL framework can be trained with more specialized application specific reward functions, such as for SAR operations in forests with thick foliage or urban regions covered in rubble in the aftermath of a natural disaster that require AGVs to specifically explore shaded regions underneath trees, boulders or buildings inaccessible to aerial surveillance using Unmanned Aerial Vehicles (UAVs). The reward for this application can be shaped as,

$$R_{SAR} = \begin{cases} 0.0075r^2 + 1.5v^2 - 0.6\omega^2 \\ 2.0 & \text{if } 2.0 < r < 2.5 \\ -50 & \text{if } r < 1.0 \end{cases} \qquad (13)$$

In addition to the reward in Equation 12, the agent is assigned a reward of 2.0 when it is between 2.0 and 2.5 m of the nearest object to promote exploration in shaded regions close to obstacles unobservable to aerial surveillance. In order to improve the safety of exploration trajectories, the agent is assigned a penalty of 50.0 when it is within 1 m of the nearest obstacle.

The focus of this paper is on evaluating the proposed DRL framework for end-to-end cognitive AGV exploration without maps, and comparing the performance of sample efficient off-policy algorithms for this task, hence we use the generalized reward function in Equation 12 for comparison analysis. The best algorithm is further evaluated using the application specific shaped reward in Equation 13 to demonstrate the effectiveness of the proposed method.

4. TRAINING AND EVALUATION

The MATLAB Robotics System [24] and Reinforcement Learning [25] Toolboxes, and Simulink are utilized to model the AGV, and train the DRL agent to compare off-policy algorithms DDPG, TD3 and SAC using the reward in Equation 12. The best algorithm is further evaluated in AutoVRL [26], an in-house high fidelity simulator developed using open-source tools for simulation to real-world DRL using the shaped reward in Equation 13.

4.1 Environments

The DRL agents are trained in two distinct environments for comparison. The first environment, depicted in FIGURE 4a, is a simple 25 m x 25 m space with walls that the agent must steer clear of. The second environment, illustrated in FIGURE 4b, is a more complex 40 m x 40 m space with walls and various obstacles, dotted in black, that the agent must additionally avoid. The trained agents from each environment are evaluated in a third environment, illustrated in FIGURE 4c, without prior training or knowledge of map characteristics to evaluate the robustness, and performance of the learned policies in a new, unknown environment with the same network architecture and hyperparameters.

The AGV, identified with a red symbol on the training maps, is set to a random starting position at the start of each training episode to enhance policy learning. This reset ensures that the agent is not biased towards any particular initial location.
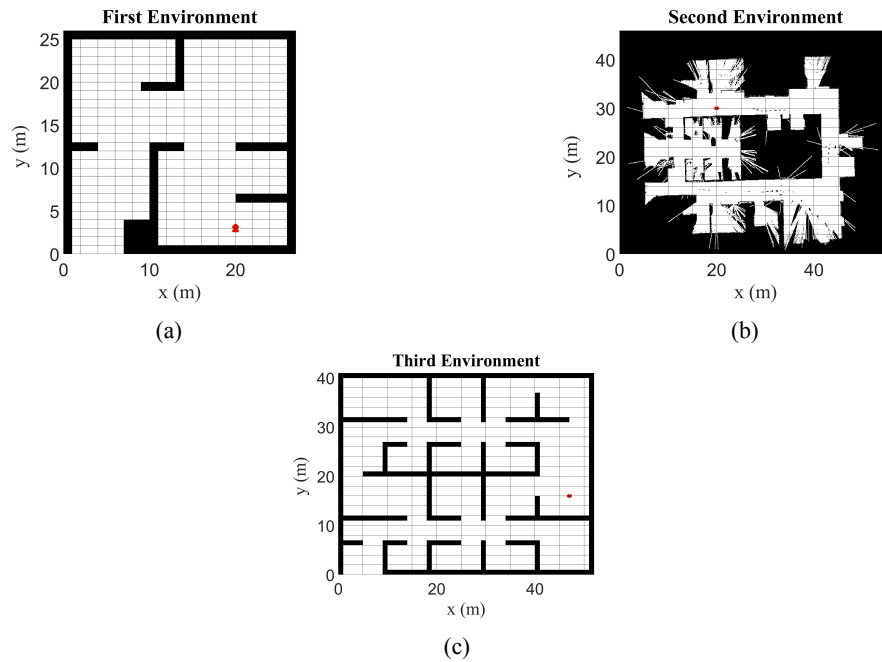
Figure 4: Training and evaluation environments with DRL agent marked in red at a randomized initial location. (a) First training environment. (b) Second training environment. (c) Evaluation environment.

To further evaluate the effectiveness of the DRL architecture for intelligent application specific exploration, a realistic 3D 20 m x 20 m outdoor environment with tree and boulder objects, illustrated in FIGURE 5, is utilized to train and evaluate the best algorithm determined from analyses of prior results.
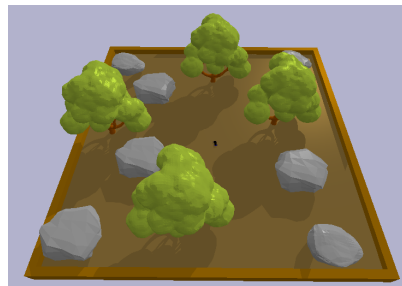

Figure 5: Outdoor environment with tree and boulder objects.

This environment includes shaded regions beneath trees and boulders unobservable to aerial surveillance, hence the shaped reward in Equation 13 is utilized to train the DRL agent to specifically explore shaded regions inaccessible to UAVs.

4.2 Exploration Quality and Efficiency Quantification

The post-training trajectories learned by the agent are evaluated for exploration quality and effi-ciency by assigning an Exploration Quality Score (EQS) and Exploration Efficiency Score (EES) to quantitatively compare the performance of each tested algorithm. The training and evaluation environments are segmented as shown in FIGURE 6, for the first training environment. Each environment is divided into 2 m^2 segments, and further segmented by circles of increasing radii with radius $10n$, where $n = 1, 2, 3, ...$
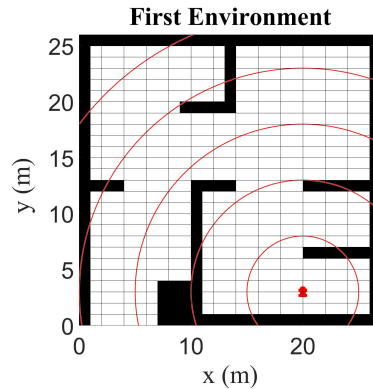

Figure 6: Occupancy grid map of first environment segmented in circles of increasing radii to score exploration quality and efficiency.

The EQS for the generated trajectory is scored as follows,

$$EQS = \Sigma_{n=1}^{n_{max}} \left(n \Sigma_{k=1}^{k_{max}} k \right) \tag{14}$$

Here k is the number of 2 m^2 segments covered by the trajectory. The EQS is greater when the trajectory covers a higher number of segments, and traverses further away from the AGV's initial position to regions encompassed by circles of larger radii. 2 m^2 segments were chosen since it is reasonable for the AGV's onboard sensors to collect viable data for down-stream applications within this region.

The EES is scored as follows where d is the trajectory distance,

$$EES = \frac{EQS}{d} \tag{15}$$

The EES is higher when the EQS is large, and the distance travelled is small. A higher EES indicates better energy efficiency, and exploration performance.

4.3 AGV Model

XTENTH-CAR [27], a proportionally scaled experimental vehicle platform, designed with similar hardware and software architectures as the full-size X-CAR [28] connected autonomous vehicle, was modeled and trained in simulation. The XTENTH-CAR AGV has a wheelbase of 0.32 *m* and utilizes the Ackermann steering mechanism.

The AGV's kinematics are computed using a bicycle model, portrayed in FIGURE 7, where the front and rear wheels are represented by a single wheel located at the center of each axle. This model is accurate for use at low speeds and offers a good balance between model accuracy and computation cost [29] for evaluation of the DRL agent.
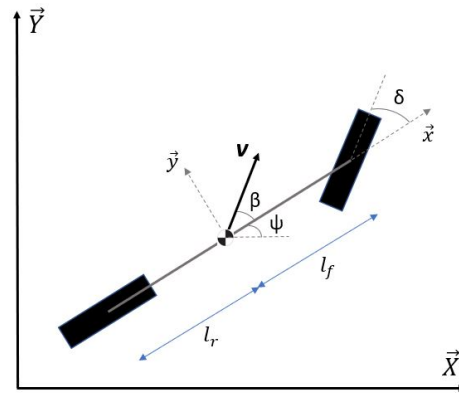
Figure 7: Schematic of kinematic bicycle model.

The bicycle model is represented by the following equations,

$$\dot{x} = v\,cos(\psi + \beta) \tag{16}$$

$$\dot{y} = v\,sin(\psi + \beta) \tag{17}$$

$$\dot{\psi} = \frac{v}{l_r}\,sin(\beta) \tag{18}$$

$$\beta = tan^{-1}\left(\frac{l_r}{l_f + l_r}\,tan(\delta)\right) \tag{19}$$

Here x and y are position coordinates of the AGV's center of mass, ψ is the angle of the AGV's heading with respect to the inertial reference frame, β is the angle between the velocity vector of the AGV's center of mass and its longitudinal axis, l_f and l_r are distances from the center of mass to the front and rear axles respectively, and velocity, v and steering angle, δ are control inputs.

4.4 Training Conditions

A training episode is concluded when the agent encounters an obstacle or completes the maximum number of steps permitted in a single episode. Subsequently, the agent is reset to a randomly determined starting location to initiate the next episode.

The DRL agent is trained to a total of 10,000 episodes, each with a maximum of 1000 steps in the first environment, and 20,000 episodes, each with a maximum of 2000 steps in the second, to facilitate rapid iterative learning. The best algorithm is trained for 5000 episodes, each with a maximum of 1000 steps in the 3D outdoor environment.

Hyperparameters that were modified with non-default values are listed in TABLE 1.

Table 1: Non-Default Hyperparameters

Hyperparameter	Value
Discount Factor (γ)	0.995
Actor Learn Rate	0.00005
Critic Learn Rate	0.0005
Target Smooth Factor	0.001
Mini Batch Size	128
Experience Buffer Length	1,000,000

5. RESULTS AND DISCUSSION

This section presents DRL training results that include post-training exploration trajectories and corresponding average return and steps achieved by the agent each episode iteration during the training period, utilizing DDPG, TD3 and SAC algorithms.

5.1 Training Performance

An Intel i7 11700K CPU and GeForce RTX 3070 Ti GPU were used for training. TABLE 2, summarizes the training times for each DRL algorithm in the evaluated environments.

SAC required the longest training time, followed by TD3 and DDPG which required the least. On average, training in the second, more complex environment required 31% longer training time than in the first, over twice the number of training episodes. DDPG required 28.5%, TD3 34.4% and SAC 30.1% longer to train in the second environment.

In the first environment, TD3 required 44.4% longer to train than DDPG, and SAC 137.4% longer than DDPG and 64.4% longer than TD3. In the second environment, TD3 required 51.1% longer to train than DDPG, and SAC 140.4% longer than DDPG and 59.2% longer than TD3. On average,

Table 2: Training Times

Agent	Training Time (Hrs)
First Environment: DDPG	66.0
First Environment: TD3	95.3
First Environment: SAC	156.7
Second Environment: DDPG	84.8
Second Environment: TD3	128.1
Second Environment: SAC	203.9

TD3 required 47.8% longer training time than DDPG, and SAC 138.9% longer than DDPG and 61.8% longer training time than TD3.

Training times ranged from 2.75 days in the first environment for DDPG to 8.5 days in the second environment for SAC. More optimal policies require longer training times to accommodate increased episode steps in the first environment, and more training episodes in the second.

5.1.1 First environment

The order 50 moving average return and agent steps during training in the first environment are illustrated in FIGURE 8 and FIGURE 9. The training results in the first environment are summarized in TABLE 3.
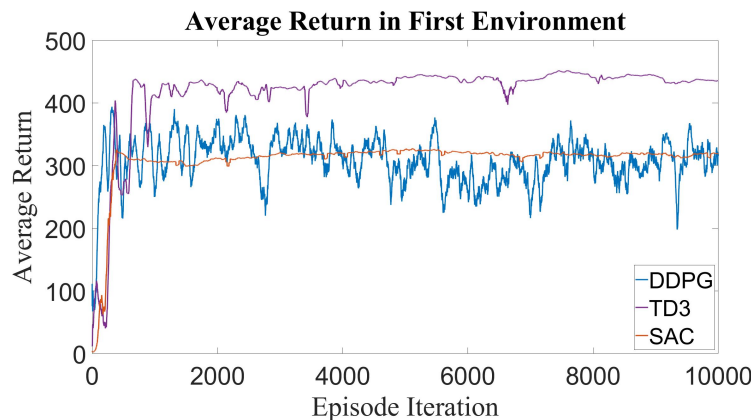

Figure 8: Order 50 moving average return during training in the first environment.

DDPG converged first at 170 episodes with an average return of 318 and 865 average steps. TD3 converged last at 960 episodes with an average return of 435 and 1000 average steps, and SAC converged at 390 episodes with an average return of 320 and the maximum 1000 average steps. TD3 achieved the highest EQS, and SAC the highest EES.

DDPG learned the least optimal policy with the lowest average return, agent steps, EQS and EES. TD3 achieves the highest return, and the maximum 1000 steps, however SAC achieves 1000 ex-
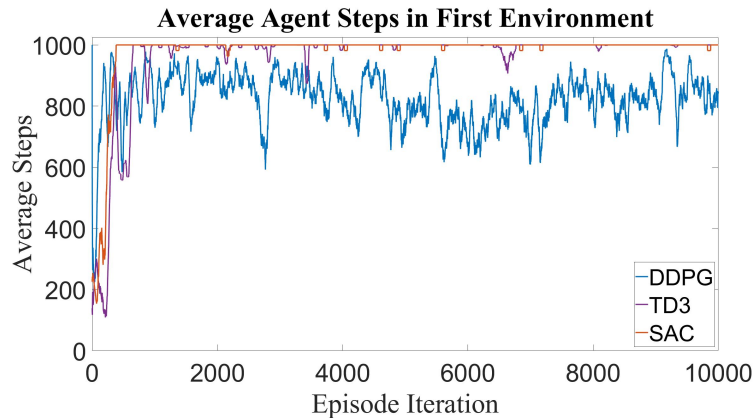
Average Agent Steps in First Environment

Figure 9: Order 50 moving average agent steps during training in the first environment.

Table 3: First Environment Training Results

Algorithm	Convergence Episode	Average Return	Average Steps	EQS	EES
DDPG	170	318	865	81	1.82
TD3	960	435	1000	**113**	2.40
SAC	390	320	1000	91	**2.63**

ploration steps more consistently post training convergence. Unlike TD3 which solely maximizes the long-term expected reward, SAC additionally maximizes the entropy of the policy to promote exploration. Consequently, TD3 learns a policy with a higher return, but SAC learns the better policy for agent exploration.

The trajectories in the first environment for each algorithm post training completion are illustrated in FIGURE 10.

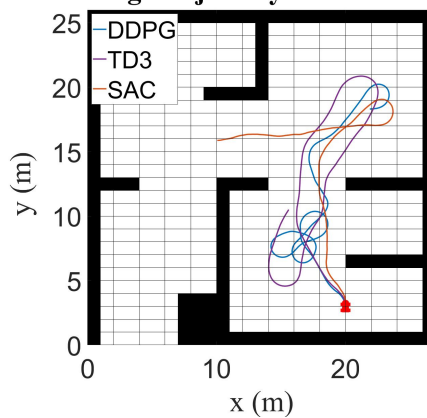**Post-Training Trajectory in First Environment**

Figure 10: Trajectories in the first environment post training completion.

Each algorithm achieved 1000 episode steps without collision. SAC covered the most ground, and exhibited the most efficient exploratory behavior which will result in the greatest energy savings. TD3 is next best, followed by DDPG which was the most inefficient, covering the same region multiple times.

5.1.2 Second environment

The order 50 moving average return and agent steps during training in the second environment are illustrated in FIGURE 11 and FIGURE 12. The training results in the second environment are summarized in TABLE 4.

Training for 20,000 episodes was insufficient for the DRL algorithms to learn an optimal policy in the second environment. At the end of the training period, DDPG achieved an average return of 125 and 530 average steps, TD3 obtained an average return of 230 and 715 average steps, and SAC converged to a local maximum at 10,620 episodes with an average return of 210 and 1050 average steps. Training was limited to 20,000 episodes to gauge performance in a reasonable time frame, however, continued training over 75,000 to 100,000 episodes will enable the agents to learn an optimal policy to traverse the more complex terrain over an indefinite number of exploration steps.
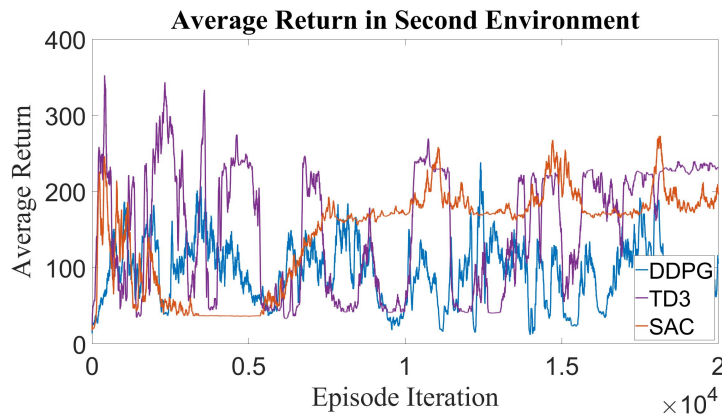

Figure 11: Order 50 moving average return during training in the second environment.

Table 4: Second Environment Training Results

Algorithm	Average Return	Average Steps	EQS	EES
DDPG	125	530	60	1.92
TD3	230	715	67	1.97
SAC	210	1050	**110**	**2.24**

Training DDPG, TD3 and SAC algorithms in the second environment for 20,000 episodes required a total of 416.8 hours, as such, it is infeasible to evaluate the algorithms for 75,000+ episodes with the existing setup. More powerful computer hardware is required. Similar to the training results
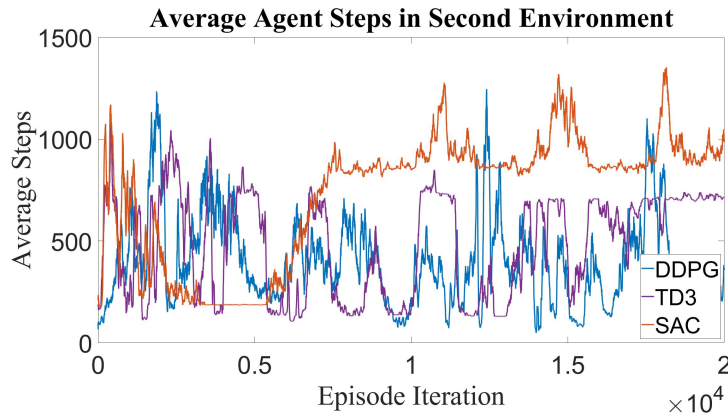
Figure 12: Order 50 moving average agent steps during training in the second environment.

in the first environment, DDPG learned the least optimal policy achieving the lowest return, agent steps, EQS and EES. TD3 achieved the highest return, however SAC learned a more optimal policy achieving the highest EQS and EES.

The trajectories in the second environment for each algorithm post training completion are illustrated in FIGURE 13.
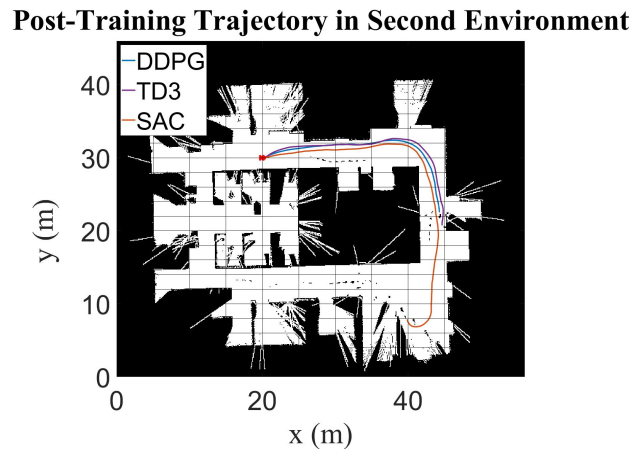

Figure 13: Trajectories in the second environment post training completion.

SAC achieved the best performance, learning a trajectory that covers the most distance. TD3 and DDPG yield similar performance, with TD3 being a marginal improvement.

5.2 Trained Policy Evaluation

The six agents, DDPG, TD3 and SAC, trained in two different environments were evaluated in a third unknown environment with no prior training or knowledge of environment characteristics, to evaluate the extensibility of the ubiquitous DRL architecture for AGV exploration in information

poor environments. FIGURE 14, portrays the trajectories for each agent in the third environment. The evaluation results are summarized in TABLE 5.
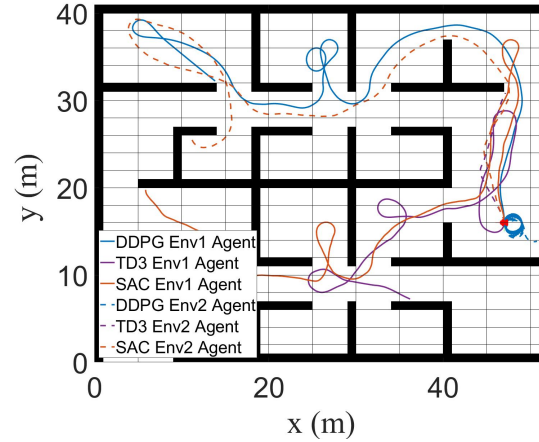
Trained Policies Evaluated in Third Environment

Figure 14: Trained DRL agents evaluated in the third environment.

Table 5: Third Environment Evaluation Results

Agent	Steps	EQS	EES
DDPG Env1	2151	252	2.36
TD3 Env1	1533	102	1.33
SAC Env1	2971	197	1.95
DDPG Env2	2893	3	0.02
TD3 Env2	321	25	1.56
SAC Env2	2839	**303**	**3.15**

The SAC agent trained in the second, more complex environment demonstrated the best performance, covering the most ground, efficiently with the highest EQS and EES. The DDPG agent trained in the first environment performed second best, covering more ground than either TD3 agent. DDPG trained in the second environment covered the second highest distance, but yielded the worst exploratory behavior with the least EQS and EES, repeatedly traversing a circular trajectory in the same vicinity. TD3 agents covered less ground, and exhibited less efficient exploratory behavior than either SAC agent. DDPG trained in the first environment performed better than that trained in the second, as the characteristics of the evaluated environment are more similar to the first than the second. The SAC agents are most robust to differences in environment characteristics, and performed better when trained in the more complex environment. Both SAC agents performed well, with the agent trained in the simple first environment achieving the third highest EQS and EES.

The reward function weights and network hyperparameters can be further engineered for this application, and the agent trained over a longer period with more episode steps each episode iteration to learn an improved policy that explores the surrounding environment indefinitely.

5.3 Shaped Reward Evaluation

SAC was utilized to learn a policy in a realistic 3D environment using the shaped reward in Equation 13 designed for AGV exploration in shaded regions unobservable to UAVs. The order 50 moving average return during training is illustrated in FIGURE 15. A post-training trajectory from a randomized initial position is illustrated in FIGURE 16.
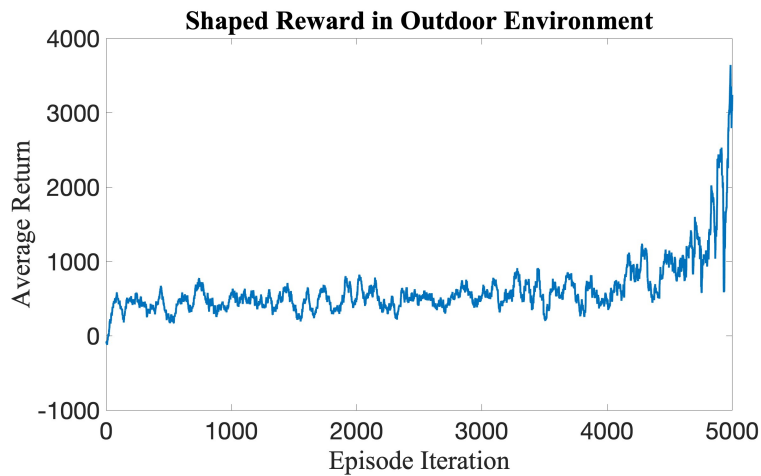

Figure 15: Order 50 moving average return during training in the outdoor environment.
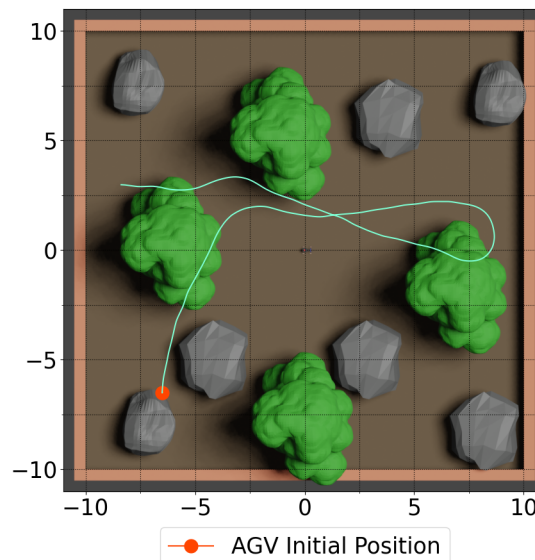

Figure 16: Trajectory in the outdoor environment post training completion.

The average return increased to 3642 at the end of the training period, yielding a policy that successfully promoted exploration in shaded regions beneath trees.

Bridging the simulation to real-world gap to transfer policies learned in simulation to real-world robotic systems is a current area of active research. The large number of episodes required to sufficiently train the agent renders simulation training an essential component for DRL in robotics applications to minimize cost and possible physical damage caused by collisions during training. Substantial computation cost is required for training, however, post-training implementation of DRL agents is significantly less expensive, which makes DRL a powerful tool for real-time AGV motion planning and control in environments without a-priori maps.

In future work, AutoVRL will be utilized to learn policies to transfer to a physical XTENTH-CAR.

6. CONCLUSIONS

This paper presented an ubiquitous DRL architecture for intelligent AGV exploration without a-priori maps. Three actor-critic DRL algorithms, DDPG, TD3 and SAC, were trained in two environments of varying complexity, and further evaluated in a third with no prior knowledge of map characteristics. Simulation results demonstrate the effectiveness of the proposed DRL architecture, reward function and training conditions for quick, efficient and collision-free AGV navigation. SAC achieved the best performance, yielding trajectories that cover the highest distance, and demonstrated the most efficient exploratory behavior. Learning requires substantial computation cost, requiring up to 8.5 days for SAC in the second, complex environment using an Intel i7 11700K CPU and GeForce RTX 3070 Ti GPU. Improved policies with higher post-training episode steps require greater training times. Despite the high training cost, post-training implementation of DRL agents is significantly less expensive, which makes DRL a powerful tool for real-time AGV exploration in information poor, dynamically altering environments. Reward shaping renders the proposed DRL framework a versatile tool for a multitude of AGV applications. SAC was trained using a reward shaped for AGV exploration in regions inaccessible to aerial surveillance in a realistic outdoor environment, yielding a trajectory that specifically explored shaded regions underneath trees, demonstrating the effectiveness of the proposed DRL framework. In future work, the simulation to real-world gap will be bridged to transfer policies learned in simulation to the physical AGV using AutoVRL, a high fidelity DRL simulator developed using open-source tools.

References

[1] Sun H, Zhang W, Yu R, Zhang Y. Motion Planning for Mobile Robots—Focusing on Deep Reinforcement Learning: A Systematic Review. IEEE Access. 2021;9:69061-69081.

[2] Victerpaul P, Saravanan D, Janakiraman S, Pradeep J. Path Planning of Autonomous Mobile Robots: A Survey and Comparison. J Adv Res Dyn Control Syst. 2017;9:1535-1565.

[3] Levinson J, Askeland J, Becker J, Dolson J, Held D, et al. Towards Fully Autonomous Driving: Systems and Algorithms. In:, IEEE intelligent vehicles symposium. Vol. IV. IEEE Publications. 2011: 163-168.

[4] Wang H, Yu Y, Yuan Q. Application of Dijkstra Algorithm in Robot Path-Planning. In: 2011 second international conference on mechanical automation and control engineering. IEEE Publications. 2011: 1067-1069.

[5]   Gammell JD, Srinivasa SS, Barfoot TD. Informed Rrt*: Optimal Sampling-Based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. In: International RSJ, editor Conference on Intelligent Robots and Systems. Vol. 2014. IEEE Publications. IEEE Publications. 2014: 2997-3004.

[6]   Vadakkepat P, Tan KC, Ming-Liang W. Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning. In: Proceedings of the 2000 congress on evolutionary computation. CEC00. Vol. 1. IEEE Publications. 2000:256-263.

[7]   Bakdi A, Hentout A, Boutami H, Maoudj A, Hachour O, et al. Optimal Path Planning and Execution for Mobile Robots Using Genetic Algorithm and Adaptive Fuzzy-Logic Control. Robot Auton Syst. 2017;89:95-109.

[8]   https://www.researchgate.net/publication/357284143_Application_of_
      Deep_Reinforcement_Learning_for_Intelligent_Autonomous_Navigation_of_
      Car-Like_Mobile_Robot

[9]   Sivashangaran S, Zheng M. Intelligent Autonomous Navigation of Car-Like Unmanned Ground Vehicle via Deep Reinforcement Learning. IFAC Papers On Line. 2021;54:218-225.

[10] Zhu K, Zhang T. Deep Reinforcement Learning Based Mobile Robot Navigation: A Review. Tsinghua Sci Technol. 2021;26:674-691.

[11] Larsen TN, Teigen HØ, Laache T, Varagnolo D, Rasheed A. Comparing Deep Reinforcement Learning Algorithms' Ability to Safely Navigate Challenging Waters. Front Robot AI. 2021;8:738113.

[12] Wijmans E, Kadian A, Morcos A, Lee S, Essa I, et al. Ddppo: Learning Near-Perfect Pointgoal Navigators From 2.5 Billion Frames. In: 8th International Conference on Learning Representations, ICLR 2020. 2020.

[13] Chen T, Gupta S, Gupta A. Learning Exploration Policies for Navigation. In: 7th International Conference on Learning Representations, ICLR 2019. 2019.

[14] Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, et al. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. IEEE Trans Robot. 2016;32:1309-1332.

[15] Ebadi K, Bernreiter L, Biggie H, Catt G, Chang Y, et al. 'Present and future of SLAM in extreme underground environments. 2022. ArXiv preprint: https://arxiv.org/pdf/2208.01787.pdf

[16] Chaplot DS, Gandhi D, Gupta S, Gupta A, Salakhutdinov R. Learning to Explore Using Active Neural Slam. In: 8th International Conference on Learning Representations, ICLR 2020.

[17] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal Policy Optimization Algorithms. 2017. ArXiv preprint: https://arxiv.org/pdf/1707.06347.pdf

[18] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, et al. 'Continuous Control With Deep Reinforcement Learning. 2015. ArXiv preprint: https://arxiv.org/pdf/1509.02971.pdf

[19] Fujimoto S, Hoof H, Meger D. Addressing Function Approximation Error in Actor-Critic Methods. In: International conference on machine learning. PMLR. 2018:1587-1596.

[20] Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, et al. Soft Actor-Critic Algorithms and Applications. 2018. ArXiv preprint: https://arxiv.org/pdf/1812.05905.pdf

[21] Jones G, Holderied MW. Bat Echolocation Calls: Adaptation and Convergent Evolution. Proc Biol Sci. 2007;274:905-912.

[22] Sutton RS, Barto AG. Reinforcement learning: an introduction. MIT press. 2018.

[23] Silver D, Lever G, Heess N, Degris T, Wierstra D, et al. Deterministic Policy Gradient Algorithms. In: International conference on machine learning. PMLR. 2014:387-395.

[24] https://www.mathworks.com/products/robotics.html

[25] https://www.mathworks.com/products/reinforcement-learning.html

[26] Sivashangaran S, Khairnar A, Eskandarian A. Autovrl: A High Fidelity Autonomous Ground Vehicle Simulator for Sim-To-Real Deep Reinforcement Learning, 2023. ArXiv preprint: https://arxiv.org/pdf/2304.11496.pdf

[27] Sivashangaran S, Eskandarian A. 'XTENTH-CAR: A Proportionally Scaled Experimental Vehicle Platform for Connected Autonomy and All-Terrain Research. 2022. ArXiv preprint: https://arxiv.org/pdf/2212.01691.pdf

[28] Mehr G, Ghorai P, Zhang C, Nayak A, Patel D, et al. X-CAR: An Experimental Vehicle Platform for Connected Autonomy Research. IEEE Intell Transp Syst Mag. 2022;15:41-57.

[29] Polack P, Altché F, d'Andrea-Novel B, de La Fortelle A. The Kinematic Bicycle Model: A Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles? In: IEEE intelligent vehicles symposium. Vol. IV. IEEE Publications; 2017: 812-818.