Online Area Covering Autonomous Robot in Unknown Environments

Guohua Ren*, Olimpiya Saha*, Javad Heydari*, Viswanath Ganapathy†, Mohak Shah
{guohua.ren,olimpiya.saha,javad.heydari,viswanath.ganapathy,mohak.shah}@lge.com
Advanced AI Lab, LG Electronics
Santa Clara, CA, USA, 95054

ABSTRACT

Autonomous area covering robots have been gaining popularity in both residential and commercial settings for a variety of purposes, including cleaning, lawn mowing, *etc.* In this paper, we explore the effectiveness of deep reinforcement learning (RL) algorithms for area coverage with minimal overlap. Through simulation experiments in grid based environments and in the Gazebo simulator, we show that Deep Q-Network (DQN) based algorithms efficiently cover unknown indoor environments. Furthermore, we demonstrate that DQN with prioritized experience replay (DQN-PER) significantly minimizes the sample complexity as well as degree of overlap when compared with DQN for area coverage. In addition, from simulations we infer that DQN-PER outperforms state-of-art online coverage algorithms, *e.g.*, BA* and Spiral-STC. Our experiments also indicate that a pre-trained RL agent can efficiently cover new unseen environments with minimal additional sample complexity. Finally, we propose a novel way of formulating an area-agnostic state representation with fixed dimensions for efficiently covering unknown environments with unknown area.

KEYWORDS

coverage path planning, reinforcement learning, autonomous robot

ACM Reference Format:
Guohua Ren*, Olimpiya Saha*, Javad Heydari*, Viswanath Ganapathy†, Mohak Shah. 2018. Online Area Covering Autonomous Robot in Unknown Environments. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Autonomous area covering robots have been deployed successfully in residential and commercial environments for several tasks [5], [8], [19]. These tasks include cleaning, lawn mowing, painting, landmine detection, surveillance and harvesting. Area covering robots aim at visiting every point in the area not occupied by the obstacles, while at the same time minimizing the overlap, coverage time, number of turns, or energy consumption [4, 11].

The coverage path planning (CPP) algorithms, based on their knowledge of the environment such as the size and shape of the area, obstacle number, and obstacle locations, can be categorized into offline and online methods. The offline CPP problem, where the robot has full geometric description of the area, is shown to be NP-hard [3]. However, many approximation or heuristic algorithms have been proposed, such as the boustrophedon or Morse decomposition based coverage algorithms [2, 7, 17], the spiral path coverage [12], and the spanning-tree based coverage [9], [10]. Further improvements in coverage have been explored using genetic algorithms [15] or using Riemannian surface properties for generic 3D environments [16].

The full-knowledge assumption is not practical, especially for commercial robots that are manufactured for general purposes such as vacuum cleaning and lawn mowing. In the online version of the problem, the robot has no, or only partial, knowledge about the size and geometry of the area to be covered, or shapes and locations of the obstacles. In such scenarios, the robot accumulates the knowledge of the environment over time using on-board sensors and data storage, and builds an online map of the area. The boustrophedon coverage, spiral path, and spanning tree techniques are adapted to the online version of the problem through using the online map [1, 6, 13, 24, 25].

The performance of the aforementioned online algorithms hinge heavily on the accuracy of the area map generated by the robot while navigating around the environment. Even though these algorithms are easily implementable on robotic platforms, they tend to achieve sub-optimal coverage with significant overlap. On the other hand, these are universal algorithms, which do not adapt to the environments the robots are being deployed. In many applications, such as vacuum cleaning and lawn mowing, the robot will be used in the same set of environments after sale. Therefore, in this paper, we employ reinforcement learning to enable the robot to adapt to the set of environments that the robot is being deployed over time.

The paper's contribution includes using DQN and its variants to achieve coverage in unknown indoor environments. Firstly, we design the state representation and a non-sparse reward function for the deep RL agent to achieve coverage with constraints on overlap. Through extensive simulations we show that the DQN agent learns to achieve coverage across environments with different area as well as number and distribution of the obstacles. Secondly, we reduce the sample complexity by employing a DQN-PER agent. We also compare the performance of the DQN-PER agent with state of the art online methods [25] and [10]. Thirdly, to account for more realistic scenarios where the area of the deployed environment is not

*First three authors contributed equally to this research.
†Viswanath Ganapathy is the corresponding author.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

available, we propose a novel way to adapt the state representation, thereby achieving an area agnostic RL agent with fixed dimensions. Furthermore, we show that this area agnostic RL agent has the advantage of being able to extrapolate coverage from a small area to a larger area without explicit training on the larger area. Finally, the performance of the RL agent is validated in the Gym-Gazebo simulator.

2 PROBLEM FORMULATION

Consider an indoor environment, discretized into equal square-shaped cells with side , generating a grid based environment of size ($=_1 \times =_2$). In our modelling, the length of the cell is equal to the diameter of the mobile robot. The unknown environment consists of multiple obstacles with varying geometry and spacing.

Recall that the objective of the area covering robot is to cover the free area of the environment as fast as possible. Minimizing the coverage time is equivalent to minimizing the number of cell revisits. To achieve this objective, we aim at finding a policy which satisfies the following constraints:

$$\min_{c} \$E4A;0? \quad \text{s.t.} \quad >E4A064\ i \quad \min\ " \tag{1}$$

In order to minimize overlap while achieving a desired coverage performance, long-term planning is required for the robot to navigate in the environment while avoiding obstacles. Hence, RL emerges as a natural solution for the area coverage task.

3 REINFORCEMENT LEARNING ALGORITHMS

Reinforcement learning handles the problem of an *agent* learning to act in an *environment*, with the goal of maximizing a predefined scalar *reward* signal.

3.1 Overview

At each discrete time step C, the agent acquires an observation $(_C$ from the environment, selects a corresponding action $_C$, then receives feedback from the environment in the form of a reward $'_{C+1}$ and the updated state information $(_{C+1}$. Such interaction is formalized as a Markov Decision Process (MDP) which is a tuple $\langle \mathcal{S}, \mathcal{A},), A, W \rangle$ where $\mathcal{S}$ and $\mathcal{A}$ contain a finite set of states and actions respectively. $)$ and A are the corresponding state transition function and reward function respectively and $W \in [0, 1]$ is the discount factor which trades off between immediate and future reward.

The way an agent selects actions is given by a policy c that defines a probability distribution over actions for each state the agent is currently at. For any time step C, we can define a discounted sum of future rewards that the agent can collect as $_C = \sum_{:=0}^{\infty} W^: \ '_{C+:+1}$. The agent aims at determining an optimal policy to maximize the expected discounted future reward.

Under a given policy c, the value of taking action 0 in a state B is given in [22] by:

$$\&_c(B,0) = \mathbb{E}[\ '_1 + W \cdot \ '_2 + """ | (_0 = B, \ _0 = 0, c] \tag{2}$$

The optimal value is $\&_*(B,0) = \max_c \&_c(B,0)$. An optimal policy can be derived by selecting the action with the highest action value in each state. Q-learning [26] has been proposed to learn the estimates of the optimal action values.

3.2 DQN and its Variants

However for most interesting problems, due to large state and/or action spaces, learning all the state action values can be intractable. To approximate the potentially high dimensional value functions $\&(B,0)$, a deep Q-network could be used: $\&(B,0;\backslash)$ with $\backslash$ being the prediction network parameters. In order to train the network, the following loss function is minimized at each iteration 8:

$$!_8(\backslash_8) = \mathbb{E}_{B,0,A,B'}[(\sim_8^{\&\#} - \&(B,0;\backslash_8))^2] \tag{3}$$

where

$$\sim_8^{\&\#} = A + W \max_{0'} \&(B',0';\backslash^-) \tag{4}$$

where $\backslash^-$ corresponds to the parameters of a separate target network. As proposed in [18], the target network shares the same structure as the prediction network except that its parameters are replicated from the prediction network every g steps, so that $\backslash^- = \backslash_8$ and kept fixed in all other steps.

Another important ingredient introduced in [18] is the use of experience replay [23]. Observed transition tuples $\langle B,0,A,B' \rangle$ are stored in a memory buffer and later sampled uniformly to update the network. A combination of the target network and the experience replay improves the DQN performance dramatically [18].

DQN samples state action transitions uniformly from the experience replay buffer. However, some experiences might be more useful than the others when training the agent. Ideally, we want to sample those experiences with higher learning potential more frequently. [21] proposed to sample transitions with probability $?_C$ relative to the difference between the prediction and the TD target:

$$?_C \propto \| \ '_{C+1} + W \max_{0'} @_\backslash^-((_{C+1},0')) - @_\backslash((_C, \ _C)) \|^l \tag{5}$$

where l is a hyper-parameter which determines the shape of the probability distribution. This variant is known as DQN with prioritized experience replay (DQN PER). Specifically, in this paper, for DQN-PER we apply Dueling Double DQN with prioritized experience replay.

Rainbow-DQN [14] has combined some of the best approaches to improve DQN like double Q-learning, prioritized experience replay (PER), dueling networks, multi-step learning, distributional reinforcement learning, and noisy nets. Rainbow-DQN has demonstrated superior performance in comparison to other DQN variants in several games of Atari.

4 DQN AGENT FOR COVERAGE IN UNKNOWN ENVIRONMENTS

In this section, we describe the DQN agent and architecture of the deep learning network for achieving efficient coverage in unknown indoor environments. The key components which influence the performance of the DQN agent are the reward function and the state representation. The goal is to train the DQN agent to achieve a predefined level of coverage in unknown environments and reduce revisits to already covered regions in the environment.

4.1 Reward Function

We experimented with several possible reward functions and came up with the following simple non-sparse function. The reward function for the DQN agent depends only on the current state: for visiting an uncovered cell, it receives a positive reward of +1, whereas for visiting a covered cell, it receives a negative reward of −0″5. Each episode will end when either the predefined coverage level is reached or the number of the steps exceeds a threshold.

4.2 State Representation

The state representation for the environments with known and unknown area is described below:

4.2.1 Area of the environment is known. The dimensions of the state is equal to the area of the indoor environment. We stack 3 matrices of size $=_1 \times =_2$ into an $=_1 \times =_2 \times 3$ tensor. The first matrix records the cells that have been covered so far. The second matrix represents the obstacle locations in the locality of the robot detected by the camera or 3D sensors. Specifically, its entries are all zeros except in the 3×3 sub-matrix of its surrounding. The third matrix captures the location of the robot, i.e., it is all zeros except for the cell that the robot is present, which is set to 1.

4.2.2 Area of the environment is unknown. We choose a dimension $= \times =$ which is smaller than the area of the environment ($= \ddot{Y} =_1 \bullet = \ddot{Y} =_2$). We stack 4 matrices of size $= \times =$ into an $= \times = \times 4$ tensor. The first matrix records the cells that have been covered so far. The second matrix represents the obstacle locations in the locality of the robot detected by the camera or 3D sensors. Specifically, its entries are all zeros except in the 3×3 sub-matrix of its surrounding. The third and fourth matrix try to capture the actual location $[G \bullet \sim]$ in the environment by decomposing it into $[G_0 + X_G, \sim_0 + X_\sim]$. $[G_0 \bullet \sim_0]$ varies from one sub-environment to another. The third matrix captures $[G_0 \bullet \sim_0]$ while the fourth matrix represents $[X_G \bullet X_\sim]$. The number of matrices we stack to the state representation can be varied to represent even larger indoor environments.

4.3 Architecture of the RL Agent

The deep neural network consists of two convolutional layers followed by two fully-connected layers. The first convolutional layer has 16 3×3 filters and the second has 32 3×3 filters, both with a stride of 1 followed by non-linearity. The final hidden layer consists of 64 rectifier units and the output layer has 4 rectifier units corresponding to valid actions (up, down, left, right) for the agent.

We use the Adam Optimizer with mini-batches of size 32 in all experiments, learning rate is fixed as 0″001. The behavior policy during training is n-greedy with n annealed linearly from 1 to 0.1 over the first 100• 000 steps, and fixed at 0.1 thereafter. Discount factor is fixed at 0″9. Huber loss is used when training our DQN agent.

5 EXPERIMENTS

5.1 Evaluation Methodology

Grid based maze environments with varying coverage area and distribution of obstacles are considered to validate the performance of the RL agent to achieve coverage with overlap constraints. We train the agent in a set of unknown indoor environments and test it in another set of unknown environments. The performance measures achieved coverage, overlap and sample complexity, which form the basis for validation of the RL agent.

The validation in the maze environments is followed by extensive experiments in the Gazebo 7 simulator using a simulated Turtlebot3 waffle pi robot. We perform our experiments in 4 simulated indoor environments with a dimension of 5×5 sq. meters. Figure 4 illustrates our Gazebo environments, the choice of which is inspired by spaces in home and office environments [20].

5.2 Simulations in Maze Environments

The experiments in the maze environments have been divided as follows to validate the performance in indoor environments.

5.2.1 Performance analysis of the RL agents. RL agents are trained on a number of environments with different sizes and obstacle configurations. Figures 1 (a) and (b) depict the performance in 15×15 and 19×19 mazes respectively as shown in Figure 3. Here we compare the performance of a dueling double DQN-PER agent and the DQN agent. For DQN-PER, we set $U = 0″6$ and $V_0 = 0″4$ where V varies linearly from V_0 to 1 during training [21].

5.2.2 Comparing RL agents with BA and Full Spiral-STC.* In order to evaluate the advantage of our proposed RL-based coverage method over classical non-learning based coverage techniques, we have selected two non-learning based coverage algorithms- Full Spiral STC [10] and BA*[25]. Full Spiral STC is a spanning tree based coverage algorithm where the robot incrementally constructs a spanning tree of the environment as it progressively completes its coverage task while maintaining either a cloockwise or anticlockwise direction of motion. BA* is an online complete coverage path planning algorithm which combines A* search with boustrophedon or zigzag motion to achieve complete coverage in unknown environments. Both these algorithms can work without an a *priori* map of the environment and have been demonstrated to achieve good coverage performance with low overlap in a variety of 2D grid environments. We have compared the extent of overlap of all the three methods, after the agent has covered 90% of the free space in the grid environments as illustrated in Figure 3. Figure 2 illustrates the comparative overlap performance achieved by all the three methods in the same grid environments. As can be observed from the figure, in all the environments, DQN-PER was able to achieve much lower overlap performance than Full Spiral STC, thus indicating its superiority over the latter. On the other hand, it can be visualized from the figure that in 4 out of 6 environments, DQN-PER was able to achieve lower overlap when compared to BA*. The lower overlap achieved by BA* in environments as shown in Figures 3 (a) and (c) can be attributed to the relatively simple nature of these environments in terms of obstacle shape and distribution. However, when the overlap performance was tested in complicated and larger environments as illustrated in Figures 3 (b), (d), (e) and (f), DQN-PER surpassed BA*.

5.2.3 Performance of RL agents in unseen environments. We validate the performance of a model, pre-trained on mazes in Figures 3 (c) and (e), in unseen mazes of the same size. Exploration rate is

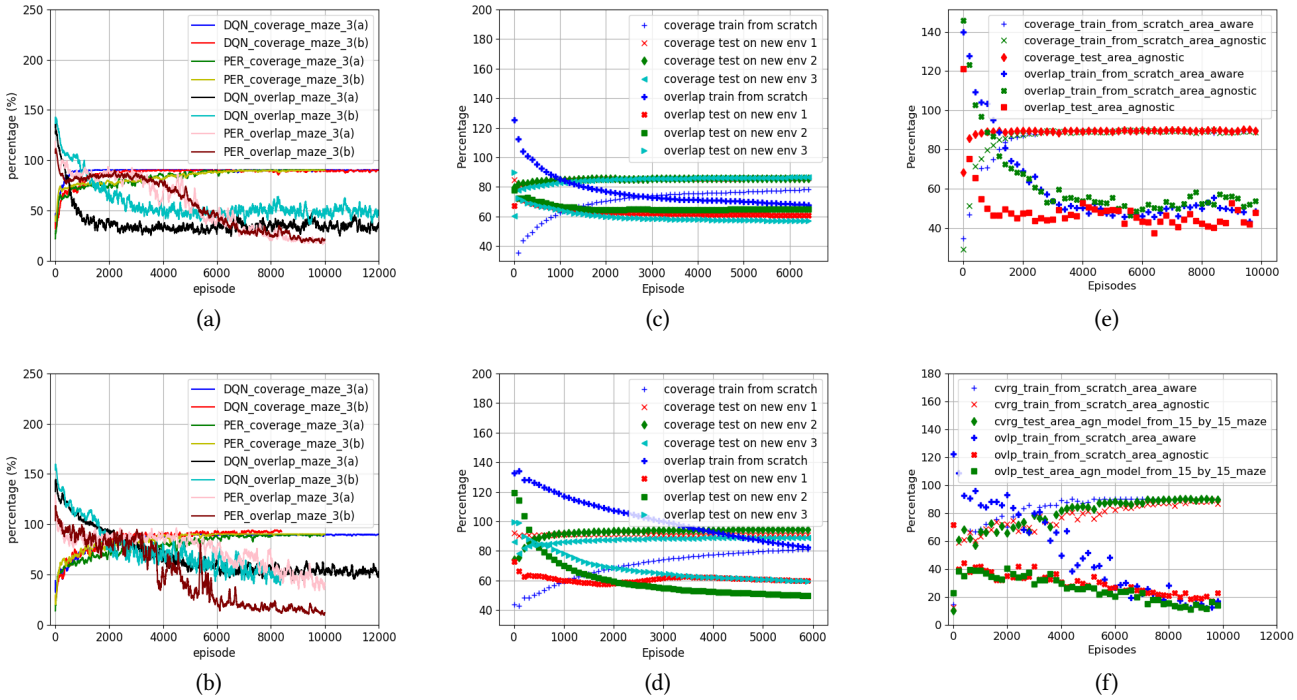


Figure 1: RL performance on (a) 15 **by** 15 **mazes and (b)** 19 **by** 19 **mazes. Transfer learning results on (c)** 17 **by** 17 **mazes and (d)** 19 **by** 19 **mazes. Area agnostic results on (e)** 15 **by** 15 **mazes and (f)** 19 **by** 19 **mazes.**

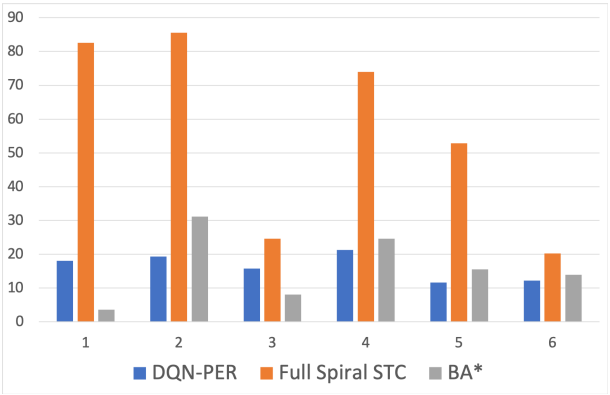


Figure 2: Comparative overlap performance of DQN-PER, Full Spiral-STC and BA* in mazes (a)-(f) from Figure 3

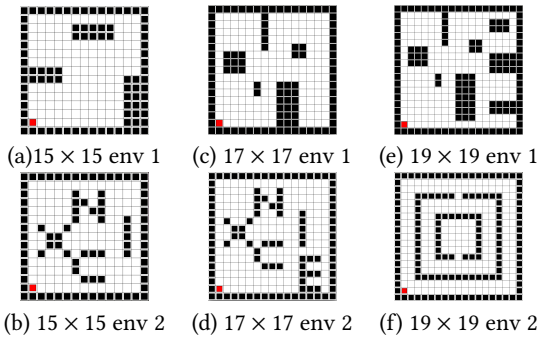
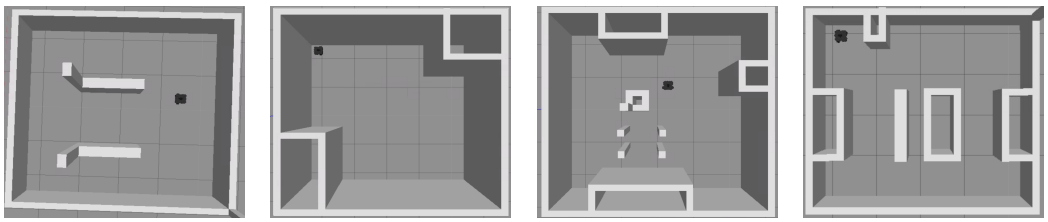


Figure 3: Example maze environments for experiments. White cells represent free space. Black cells indicate static obstacles.

fixed at 0″1 all the time. Figures 1 (c) and (d) show that for environments with different sizes, using a pre-trained model achieves faster convergence, leading to a higher level of coverage with significantly reduced overlap. Furthermore, by using pre-trained model weights, higher initial coverage is achieved in testing environments. The ability of pre-trained models to achieve high initial coverage in unseen environments is attractive for practical deployment in real world area covering robots. Factory trained robots will be able to perform at a high initial coverage and improve the coverage efficiency with additional samples from the deployed environment.

5.2.4 Area agnostic RL agent. So far, we have trained and evaluated our DQN agents and achieved targeted performance across varying environment sizes. However, in real life scenarios where room areas vary and layouts are complicated, a DQN agent trained with state representation proportional to the area of the environment will fail to perform adequately. This offers motivation for training an agent which is area agnostic.



(a) Env 1 (b) Env 2 (c) Env 3 (d) Env 4

Figure 4: Indoor environments for our simulated experiments.

The state representation for environments with unknown area is modified as described in section 4.2.2. The modified state representation allows the location of the robot in environment to be described in terms of an offset and relative position from the offset. Therefore, the combined state representation for unknown area is similar to the state representation for environments with known area. The locations of the cells visited by the agent is retained in the DQN memory buffer. In this way, the area agnostic DQN agent learns to cover the entire area based on state representation of the multiple intersecting smaller sub-environments.

An area agnostic agent (= = 13) is trained on a 15 by 15 environment in Figure 1 (e), and achieves desired coverage. Also the sample complexity of area agnostic agent is comparable to the area aware agent (= = 15). Testing is performed on a new environment using the trained area agnostic DQN agents. Similar to the area aware scenario, higher initial coverage and faster convergence are observed in Figures 1 (e).

Similarly for a 19 by 19 environment, an area agnostic agent (= = 13, green curves) achieves comparable performance as an area aware agent (= = 19, blue curves). We also evaluate how the agnostic agent (= = 13) trained on the 15 by 15 environment in Figure 3 (d) performs on the 19 by 19 environment in Figure 3 (f). As shown in Figure 1 (e) and (f), the model trained in a small environment generalizes well to an unknown environment with a larger area, without increasing the sample complexity. Learning to cover a larger 19 by 19 area based on an area agnostic agent trained on an unrelated 15 by 15 environment points to a promising direction for RL model deployment in real world robots.

5.3 Simulations in Gazebo

We have conducted extensive experiments in the gym-Gazebo simulator to validate the performance of learning-based algorithms in situations closely simulating the real-world.

In this study we conduct a comparative performance analysis of DQN and Rainbow-DQN in the context of area coverage. Our experiments are divided into two sets. In the first set of experiments, we train the RL agent from scratch in each environment and analyze its performance. In the second set of experiments, we use the pre-trained weights achieved by training the RL agent in one of the environments and train it in a different environment. The first set of experimental results helps us analyze the general efficiency of the RL agent for the coverage task whereas the second set of experimental results helps us assess the generalization capability of the RL agent when deployed in a new environment after having been trained.

Figures 5 (a)-(d) illustrate the performance of DQN and Rainbow-DQN in the 4 environments from Figure 4. We can see that overall Rainbow-DQN achieves lower overlap and better or comparable average coverage in all environments. In environments 1, 3 and 4, Rainbow-DQN and DQN were able to achieve a final average coverage close to 80% while achieving maximum overlap of 40%. In environment 2, coverage performance achieved by DQN and Rainbow-DQN without deterministic action selection were close to 60%. The lower coverage performance in environment 2 can be attributed to the wide open spaces in this environment in contrast to the other two environments which are cluttered with multiple obstacles.

Figures 5 (e) and (f) demonstrate the performance of DQN and Rainbow-DQN in environment 3 using a pre-trained network which achieves coverage in environment 1 (Figure 4). The pre-trained model enables the RL agent to start with a higher initial coverage as well as lower initial overlap.

6 CONCLUSIONS

In this paper we explored the possibility of employing an RL agent for the area coverage task and demonstrated that the DQN-PER based RL agent achieved coverage with lower overlap when compared with well known online coverage algorithm, BA* and Spiral-STC . Furthermore, RL agents based on DQN-PER substantially reduced the sample complexity, making it amenable for deployment on real-world robots. Our experimental studies in the grid based environments as well as Gym-Gazebo simulator have demonstrated the practicability while deploying trained RL agents in unseen environments. Taking into consideration the fact that information regarding the area of the environment may not be available, we made further modifications to the state representation which led to an area agnostic DQN agent. The ability of the RL agent to generalize across different obstacle configurations was validated through an extensive series of experiments on both grid based maze environments and gym-gazebo simulator. The results in the maze and Gym-Gazebo environments achieved similar performance with regard to sample complexity, average coverage and overlap.

In our future work, we aim to validate the performance of the learning-based algorithms on a physical hardware robot as well as extend these algorithms to environments with dynamic obstacle(s) and enable the robot to manage battery charging based on the power status.