Exploration of Unknown Environment using Deep Reinforcement Learning

Asad Ali[1*], Sarah Gul[1], Tallat Mahmood[2] and Anayat Ullah[2]
[1]Control, Automotive and Robotics Lab, National Centre of Robotics and Automation (NCRA), Rawalpindi, Pakistan
[2]Department of Electrical Engineering, Balochistan University of IT, Engineering and Management Sciences, Quetta, Pakistan
*email: asadaligondal0007@yahoo.com

2023 International Conference on Robotics and Automation in Industry (ICRAI) | 978-1-6654-6472-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRAI57502.2023.10089589

***Abstract*—Exploring the unknown environment is a very crucial task where human life is at risks like search and rescue operations, abandoned nuclear plants, covert operations and more. Autonomous robots could serve this task efficiently. The existing methods use uncertainty models for localization and map building to explore the unknown areas requiring high onboard computation and time. We propose to use Deep Reinforcement Learning (DRL) for the autonomous exploration of unknown environments. In DRL, the agent interacts with the environment and learns based on experiences (feedback/reward). We propose extrinsic and curiosity-driven reward functions to explore the environment. The curiosity-based reward function motivates the agent to explore unseen areas by predicting future states, while the extrinsic reward function avoids collisions. We train the differential drive robot in one environment and evaluate its performance in another unknown environment. We observe curiosity-driven reward function outperformed the extrinsic reward by exploring more areas in the unknown environment. The test results show the generalization capability to explore unknown environments with the proposed methods.**

***Keywords*—Double Deep Q-Network, Exploration, Mobile Robots, Navigation, Reinforcement Learning**

I. INTRODUCTION

Exploration is crucial for many applications where human operations are infeasible. Autonomous exploration can benefit the search and rescue team to locate humans in disastrous conditions efficiently to explore abandoned buildings, nuclear plants, and coal mines. Exploration is also required in covert operations. Bio-inspired robots are progressing in environment exploration research [1] due to agile structure. Deep Reinforcement Learning (DRL) methods have extensively been studied for learning exploration policies for mobile robots. Deep neural networks have a very powerful representation ability compared to traditional approaches involving intermediate steps such as Simultaneous Localization and Mapping (SLAM) [2]. SLAM algorithms use a series of sensor readings to concurrently map the surroundings and estimate the location of the robot sensor with the underlying presumption of a predetermined and predictable sequence of control instructions or actions, such as velocity instructions for a robotic vehicle that carries the sensor. In recent years several DRL works have shown the possibility of learning navigation policies directly from raw sensory inputs, bypassing the need for localization, mapping, and path-planning methods. Recent works involve successor feature Reinforcement Learning (RL) for transferring navigation policies [3], and multi-robot collision avoidance [4].

Mapless path planning solutions caught scientists' interest, particularly in the last ten years, due to recent advances in RL [5]. Successful RL-based solutions for navigation and exploration are put forward in [6], [7]. However, training time and data requirements are sometimes lengthy for these mapless path planners to work successfully.

This paper focuses on exploration methods of learning policy in which the agent tries to explore as much area as possible during training. The mentioned methods mostly use exploration policies that rely on pure randomness, like epsilon greedy. Moreover, in real world the reward to the agent is sparse. In most of the RL problems there is a goal to reach in the environment and the reward function is modeled based on the goal. The problem of exploration and navigation becomes more complex when there is no goal all together and the agent has to explore maximum area. Environment exploration by the methods mentioned in the former paragraph becomes challenging in complex and sparse reward environments having sharp turns and long corridors. Due to this, these methods limit the mobile robots' learning exploration and navigation policies.

Defined by [8], the intrinsic curiosity of the agent is the ability of the agent to predict the consequences of its actions in the environment by giving information about the uniqueness of states. This idea is formally known as the Intrinsic Curiosity Module (ICM). In [8], using video games as a test bed for ICM, they showed improved results compared to the external reward. ICM can be used during RL as an intrinsic reward for the agent. This reward formulation helps the agent to explore novel states. One of the similar intrinsic reward formulations is used in [9], in which they formulated intrinsic curiosity as a transition model for better exploration policy to take action.

To learn better policy in complex settings that may have dead ends, we specifically study mobile robot exploration techniques for DRL agents in this work. We also test the

978-1-6654-6472-7/23/$31.00 ©2023 IEEE

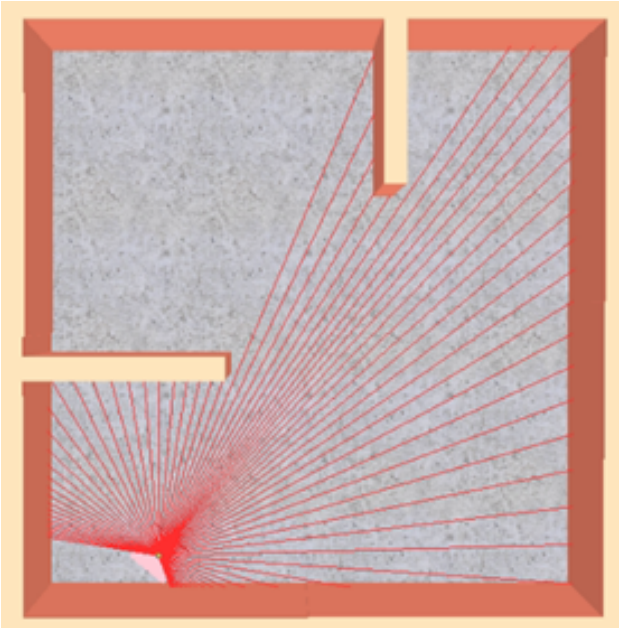

Fig. 1. This Map-1 is used for training the agent, created in a Simulation environment constructed in CoppeliaSim (formally known as V-REP), where a differential drive robot is our agent. The agent avoids the obstacles and explores the environment as its objective.

generalization capability on a different map.

II. BACKGROUND

A. Reinforcement Learning Framework

Any decision-maker is considered an agent in RL, and everything around the agent is referred to as the environment. The agent receives a numerical value $R \in \mathbb{R}$ called a reward as a feedback signal by interacting with the environment. The agent seeks to increase the cumulated reward. The agent's objective is to learn the actions that, when it starts from a certain start point, takes actions, and finally reaches a certain end point, will maximize the long-term expected return. To represent the return value, we may introduce a discount factor $\gamma \in [0, 1)$ the return value can be represented as:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots = \sum_{k=0}^{\inf} \gamma^k R_{t+k+1} \quad (1)$$

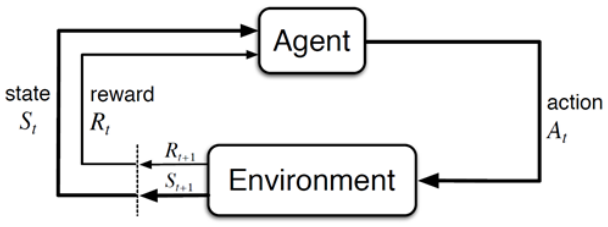

Fig. 2. Agent-environment interaction. S_l is the state of the agent at timestep t in the environment; it takes action A_t based on the observation and moves to the next state S_{t+1} and in the process it receives a reward R_{t+1}.

The Fig. 2 illustrates the agent-environment interaction in RL settings. In a generic RL problem, an agent interacts with its environment at each time step through a discrete-time stochastic process. At any timestep t, in the environment's state $S_t \in \mathcal{S}$ ($\mathcal{S}$ is a finite set of states) the agent selects an action $A_t \in \mathcal{A}$ where $\mathcal{A}$ is a finite set of actions, as the agent takes action, it receives a reward, $R_{t+1} \in \mathbb{R}$ and moves to the new state S_{t+1}. Where S_t and A_t represents defined discrete probability distributions of random variables.

B. Q-Learning

Finding the best estimates for the actions in terms of the expected future reward R_t, can solve a sequential decision problem. The value of an action by following a policy denoted by π in a given state, $Q_\pi(s, a)$ is defined as:

$$Q_\pi(s, a) \doteq E_\pi[R_t | S_t = s, A_t = a], \forall s \in \mathcal{S}, a \in \mathcal{A} \quad (2)$$

The action values are learned using Q-learning in [9]. Most of the robotic complex problems require action values for all states. The term $Q(s, a; w)$ may be used as a parametrized value function, here, w denotes the parameters. Action A_t is taken while in the state S_t, moving to a new state S_{t+1}, and obtaining a reward R_{t+1}, then the parameter values in Q-learning are updated following Eq. 3:

$$\begin{aligned} w_{t+1} = w_t + \alpha(R_{t+1} + \gamma \underset{a}{max} Q(S_{t+1}, a; w_t) \\ -(S_t, A_t; w_t)) \nabla_{w_t} Q(S_t, A_t; w_t) \end{aligned} \quad (3)$$

The learning rate is represented byα. The target is defined as Eq. 4:

$$Y_t^Q = R_{t+1} + \gamma \underset{a}{max} Q(S_{t+1}, a; w_t) \quad (4)$$

The target value is the reward plus a discounted maximum future Q value. The Q-learning algorithm has an overestimation problem as it chose greedy values.

C. Double Q-Learning

In [11], the double Q-learning technique is described as a solution to the Q-learning overestimation issue. It divides the target's maximum operation into action selection and evaluation. In double Q-learning algorithm, the update of one of the value functions is random, this allows the learning of two value estimation functions from Experience Replay Memory (ERM). There are two sets of weights: w_t and w'_t, one computes the greedy policy for each update, and the other computes its value. Double Q-learning has a target:

$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{argmax} Q(S_{t+1}, a; w_t); w'_t) \quad (5)$$

The weights w_t in the argmax are used to select an action in Eq. 9. Q-learning utilizes a greedy policy for action estimation, but in double Q-learning, a separate set of weights w'_t are used to evaluate this greedy policy. By switching the functions

of w_t and w'_t, the weights w'_t are updated iteratively after some time. Combining non-linear function approximation with double Q-learning, the model-free RL technique double Q-learning avoids overestimation, leading the Q-learning process to diverge as there is a correlation between samples and non-stationary targets [12]. In [13], Deep Q-Network (DQN) is presented to handle this divergence.

D. Deep Q-Network

DQN has significantly advanced, and the DQN artificial intelligence agent has been introduced in [13]. For a given state S_t, DQN produces an estimate of Q-values $Q_\pi(S_t, a; w)$ called state-action values, with w representing the network weights. The neural network is a function approximator that maps states to actions. The DQN uses ERM and a fixed target network to address the divergence issue with Q-learning. The correlation problem is addressed in the ERM. After a specific number of iterations, fixed goal parameters or weights w_t^{tar} are utilized for the computation of target values, and the weights are updated with w_t to improve the stability issue. The DQN target is defined as:

$$Y_t^{DQN} = R_{t+1} + \gamma \underset{a}{max} Q(S_{t+1}, a; w_t^{tar}) \tag{6}$$

Eq. 6 shows that the deep Q-learning algorithm overestimates the values. The weights are updated continuously by sample selection from the replay memory. The ERM consists of the agent's experiences.

E. Double Deep Q-Network

While the DQN manages the divergence problem, the over-estimation problem is handled by double Q-learning. A novel approach has been described in [14] by merging Double Q-learning with DQN, referred to as Double Deep Q-Network (DDQN). The online network and the target network are identical, both are used by the DDQN. The target network of the DDQN is used to evaluate the greedy policy while evaluating it through the online network. The target of DDQN is represented in Eq. 7:

$$Y_t^{DoubleDQN} = R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{argmax} Q(S_{t+1}, a; w_t); w'_t) \tag{7}$$

The DDQN evaluates the action quality using the target network parameters w_t^{tar}. Using the w_t weights, the greedy policy is estimated by the Q-network. The parameters of the target network are changed with the weights of the second network w_t to compare the current greedy policy to the Double Q-learning (Eq. 5). The update to the target network contains a recurring copy of the online network and is the same as the one for the DQN.

III. METHODOLOGY

A. Framework

To achieve the exploration task, we define the environment with walls and obstacles depicting a real-world scene, as illustrated in Fig. 3. A differential drive mobile robot is equipped with LiDAR as the agent.

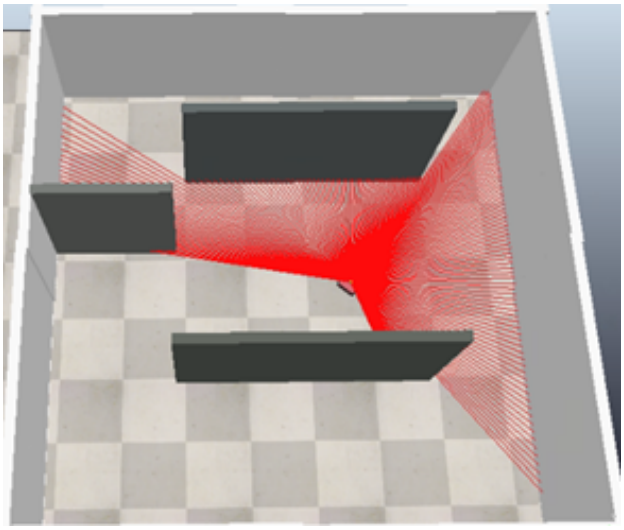

Fig. 3. Testing map (Map-2), the agent previously trained is tested on this map to test the generalization capability of the learned policy on Map-1.

B. State Space

State space is continuous in this setup. The 200-dimensional laser range readings are taken at any timestep as S_t. The LiDAR scans are the relative distance between the agent and the environment.

C. Action Space

The action space is discrete in this proposed framework, and the agent can take three actions (move forward, rotate left and right). The actuating signals to the wheels of the mobile robot are given in the form of left and right wheel velocities. The difference between two-wheel velocities achieves the effect of moving left, and right. To move left, the wheels receive $(v_{left} = 0, v_{right})$, where v_{left} refers to the left wheel velocity and v_{right} right wheel velocity. To move right, the wheels receive $(v_{left}, v_{right} = 0)$, and to move forward, (v_{left}, v_{right}). We pass the same non-zero value for v_{left} and v_{right} to achieve the actions in $\in \mathcal{A}$.

D. Extrinsic Reward

A reward function is formulated to optimize a dynamic optimization problem via RL. Often, this reward function only includes some defined cases. This limits the power of RL to achieve a generalized solution. To learn the exploration policy, we formulate two different settings with the only difference of reward functions. In this work, we propose two reward functions, extrinsic reward and intrinsic reward. The intrinsic reward is elaborated in section III-E. We define this traditional RL reward function as an extrinsic reward $R_t^{extrinsic}$ from the environment at timestep t as:

$$R_t^{extrinsic} = \begin{cases} -100 & \text{If collided} \\ 0 & \text{Otherwise} \end{cases} \quad (8)$$

The $R_t^{extrinsic}$ in Eq.8 motivates the agent to explore the environment while avoiding obstacles. The agent is penalized upon collision with walls or obstacles, and the episode terminates. An episode is one round of exploration. Everywhere else, the agent receives a reward of 0, which is better than -100. The extrinsic reward $R_t^{extrinsic}$ is measured to check the learning trend of the exploration task.

E. Intrinsic Reward Formulation

To overcome the limitations of extrinsic reward formulation, we propose a curiosity-based reward function that motivates the agent to learn to explore the environment's unvisited areas. With $R_t^{extrinsic}$ reward, we augmented the total reward with the intrinsic reward $R^{intrinsic}$, which encourages the agent to reward novel states more, consequently improving the generalization and learning of the DRL agent.

We formulated the intrinsic reward $R^{intrinsic}$ following the formulation of [8]. The curiosity module consists of a froward model:

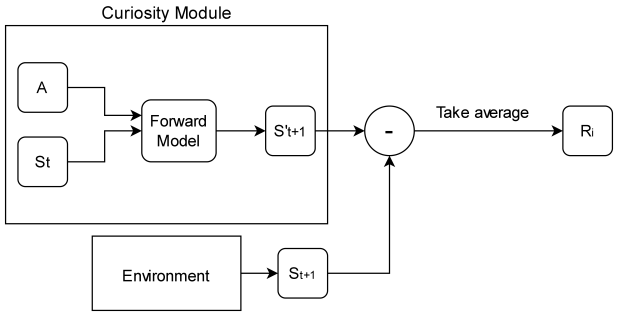

Fig. 4. ICM architecture. Forward model takes the action A_t and state S_t as inputs and infer the next state S'_{t+1}, then the difference between S'_{t+1} and S_{t+1} is computed as $R^{intrinsic}$.

The Fig.4, illustrates the formulation of $R^{intrinsic}$. It consists of a neural network (forward model). The inputs of the forward model are the current stateS_t and action A_t, and it predicts the next state S'_{t+1}. The mean value of the prediction error between S'_{t+1} and S_{t+1} is taken as the intrinsic reward $R^{intrinsic}$:

$$R^{intrinsic} = \sum \frac{(S'_{t+1} - S_{t+1})}{n} \quad (9)$$

In Eq.9, the n represents the dimensions of LiDAR scans. The $R^{intrinsic}$ reward motivates the agent to visit novel states in the environment; hence, the agent avoids learning suboptimal policy for exploration and local minimums.

F. Neural Net Architecture

The DDQN architecture consists of four fully connected layers of 200, 64 and 64 neurons followed by ReLU activation function and the last layer of 3 (size of action space) neurons followed by linear activation. The curiosity module consists of three fully connected layers having 201, 64 neurons with ReLU activation and the last layer with 200 (dimensions of LiDAR scans) neurons with linear activation. The inputs to the ICM module are the action and the current state, it predicts the next state as an output.

IV. TRAINING AND RESULTS

A. Training

The Whole formulation is implemented in coppeliasim robotic simulator. The neural network architecture defined in section III-F is used. Table. I defines the hyperparameters of the neural networks and DDQN for this work. DDQN is trained with adam optimizer having a learning rate of 0.002. After every 7 episodes, the weights of the target network are updated. The curiosity module is also trained with adam optimizer having 0.002 learning rate. We only trained the agent on *Map-1*. All the experiences (S_t, A_t, R_t, S_{t+1}, Done) stored in ERM. Random batches are sampled from the ERM for training. Training and testing are carried out in *Map-1* and *Map-2* as shown in Fig. 1 and Fig. 3 respectively. We trained the agent on *Map-1*.

TABLE I. HYPER PARAMETERS FOR BOTH INTRINSIC AND EXTRINSIC REWARD FORMULATIONS.

Parameter	Value
Hidden layers' activation function	relu
Output layer's activation function	linear
Optimizer	Adam
Discount factor (γ)	0.99
Learning rate (α)	0.002
Target network update frequency	7
Replay buffer	20000
Batch size	64
Maximum steps per episode	400
Number of training episodes	1000

B. Results

The Fig. 5 represents the training $R^{intrinsic}$ defined by Eq. 9. The reward is more at the start of the training, it drops as the agent gets familiar with the environment. The rise in $R^{intrinsic}$ curve after 600 episodes, the agent tries to increase the curiosity reward by exploring more novel areas (novel states) of the environment.

The explored area increases as the number of actions increases. Fig. 6 shows the training episode steps for both methods in $Map-1$. The blue and orange curves in Fig. 6 are the training episodic steps for DDQN with curiosity and DDQN with extrinsic reward formulation, respectively. The rise in both curves in Fig. 6 depicts the improved exploration learning of the agent as it collects more experiences. Due to the $R^{intrinsic}$, the agent explores more as compared to

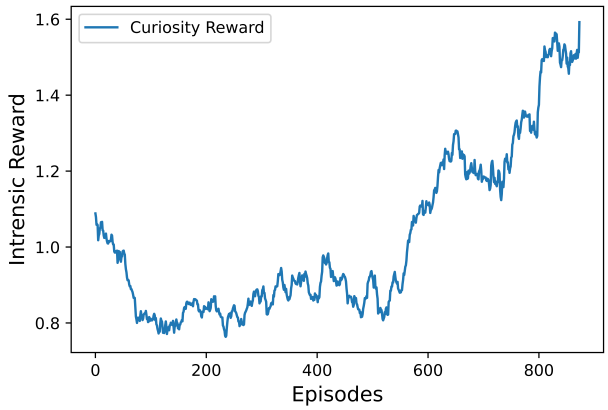

Fig. 5. Illustration of training curiosity reward $R^{intrinsic}$ in $Map-1$. The rise in the curve defines the motivation to explore the unknown areas of the environment.

the $R^{extrinsic}$ only. With a curiosity reward, the agent can explore further in the environment during training compared to the only extrinsic reward. The agent sees a bigger part of the environment.

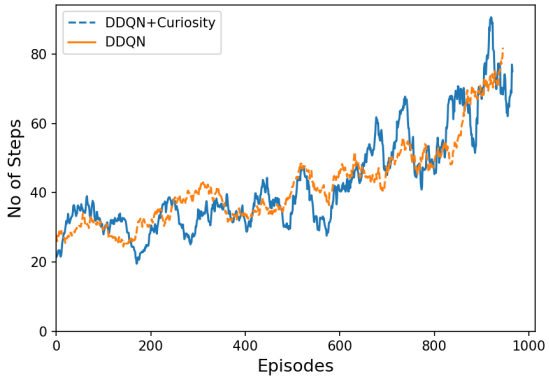

Fig. 6. Illustration of training episode reward for DDQN with $R^{intrinsic}$ (blue curve) and $R^{extrinsic}$ (orange curve), respectively. The rise in Both curves depicts the increasing exploration ability of the agent.

We define the action policies learned by DDQN with $R^{intrisic}$ and DDQN with $R^{extrinsic}$ only as $\pi_{intrinsic}$ and $\pi_{extrinsic}$, respectively. The $\pi_{intrinsic}$ and $\pi_{extrinsic}$ are the policies with the highest training episode steps, saved during the training process in $Map-1$. These policies are exploited during the test process $Map-2$. Fig. 7 illustrates the test episode steps. The blue and orange curves in Fig. 7 show episode steps in the test phase using $\pi_{intrinsic}$ and $\pi_{extrinsic}$ action policies, respectively. We observe from Fig. 7 that both proposed methods significantly explore the unknown environment $Map-2$, compared to the training phase. In the training phase, the highest episode steps are 92 for $R^{intrinsic}$ method while 79 for $R^{extrinsic}$ formulation. In the test phase, we observe maximum episode steps are 356 and 283 while minimum episode steps are 205 and 77 for $\pi_{intrinsic}$ and $\pi_{extrinsic}$, respectively. Fig. 7 benchmarks the proposed method of combining DDQN with curiosity as it outperforms the DDQN with only extrinsic reward by exploring more areas. In all test episodes, the $\pi_{intrinsic}$ gives better generalization, rewards, and steps, and the agent traverses more areas of the unknown environment.

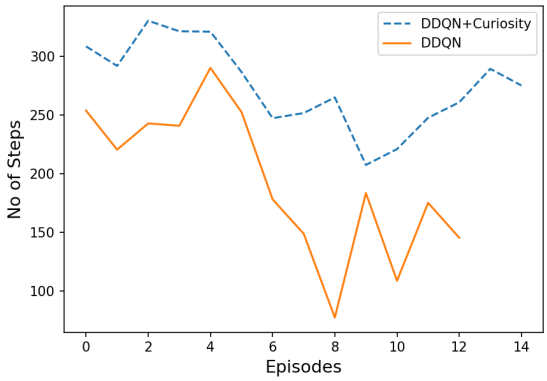

Fig. 7. Illustration of test episodes. The blue and orange curve shows episode steps using $\pi_{intrinsic}$ and $\pi_{extrinsic}$ policies on $Map-2$. It shows that $\pi_{intrinsic}$ outperforms the $\pi_{extrinsic}$ in exploring more areas.

C. Discussion

From the results, it can be concluded that the improvement in learning the policy is increased by combining DDQN and curiosity reward. The agent trained with ICM shows a better exploration of unseen maps. We bench-marked that using ICM agent explores the environment better and avoids local minima of the policy. Using ICM, agents can see a bigger part of the environment without revisiting the states, unlike the random exploration methods. The proposed method highlights the generalizability in exploration using ICM. The agent is motivated to explore novel states using the intrinsic reward and tries to avoid the parts of the environment that have been visited.

V. CONCLUSION

In this research, we presented an autonomous exploration framework for a differential drive robot. In this framework, the robot learns to explore unknown environments without any map or prior knowledge of the environment. We proposed using extrinsic and intrinsic rewards (curiosity-driven) to learn exploration policies with DRL. The inherent capability of intrinsic reward motivates the robot to explore unseen areas of the environment. The intrinsic reward learns the consequences of the actions by predicting the next state. The comparative study of intrinsic and extrinsic rewards illustrates that our presented method achieves better generalization capability in exploring novel, unknown areas of the environment. Future work will include the implementation of this framework on robotics hardware and better reward functioning weighting.

ACKNOWLEDGMENT

This research is conducted at Control Automotive and Robotics Lab (CARL-BUITEMS), funded by National Center of Robotics and Automation (NCRA) with the collaboration of Higher Education Commission (HEC) of Pakistan.

REFERENCES

[1] S. N. Khan, T. Mahmood, S. I. Ullah, K. Ali, and A. Ullah, "Motion planning for a snake robot using double deep q-learning," in 2021 International Conference on Artificial Intelligence (ICAI), 2021, pp. 264–270.

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

[3] J. Zhang, J. T. Springenberg, J. Boedecker and W. Burgard, "Deep Reinforcement Learning with Successor Features for Navigation across Similar Environments," Intelligent Robots and Systems (IROS), pp. 2371-2378, 2017.

[4] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," IEEE International Conference on Robotics and Automation (ICRA), pp. 6252-6259, 2018.

[5] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 2018.

[6] Y. Wu, Y. Wu, G. Gkioxari and Y. Tian, "Building Generalizable Agents with a Realistic and Rich 3D Environment," in arXiv:180102209, 2018. capitalized," J. Name Stand. Abbrev., in press.

[7] L. Tai, G. Paolo and M. Liu, "Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation," in International Conference on Intelligent Robots and Systems, 2017.

[8] D. Pathak, P. Agrawal, A. A. Efros and T. Darrel, "Curiosity-driven Exploration by Self-supervised Prediction," International Conference on Machine learning (ICML), 2017.

[9] N. Watters, L. Matthey, M. Bosnjak, C. P. Burgess and A. Lerchner, "Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration," arXiv preprint arXiv:1905.09275 , 2019.

[10] C. John and C. H. Watkins, Learning from delayed rewards, United Kindom, 1989.

[11] H. v. Hasselt, "Double q-leaming," Advances in neural information processing systems, vol. 23, pp. 2613-2621, 2010.

[12] V. Mnih, K. Kavukcuoglu, . D. Silver, A. Graves, . I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," NIPS Deep Learning Workshop, December 2013.

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

[14] H. Van Hasselt, A. Guez and D. Silver, "Deep reinforcement learning with dounle Q-learning," AAAI Conference on Artificial Intelligence, 2015.