Reinforcement Learning-based DDPG and PRM Integration for Optimal Navigation in Varying Complex Environments

Dr. E. Madhusudan Raju
Department of Mechanical Engineering
University College of Engineering
Osmania University, Hyderabad
Telangana, India
madhusudhan.e@uceou.edu

Cheruku Nikhil
Department of Mechanical Engineering
University College of Engineering
Osmania University, Hyderabad
Telangana, India
Nikhilchrk@gmail.com

Chanda Joshith
Department of Electrical Engineering
University College of Engineering
Osmania University, Hyderabad
Telangana, India
joshithc.be25@uceou.edu

***Abstract*—We have Autonomous navigation in various environments is critical for real-world applications. Adaptation to new environments and complexities is crucial for autonomous mobile robots. This project investigates feasibility of finding the optimal sample time and average reward structure for a given size of environment using Reinforcement Learning based Deep Deterministic Policy Gradient (DDPG) technique. This project finds the optimal connection distance and number of nodes to be given in finding the feasible path using Probabilistic Roadmap (PRM). Integrating these two techniques will enable a mobile robot to avoid obstacles while reaching the destination by following an optimal path. As a case study a mobile robot is trained using DDPG in three different environments of size 26x27 units with varying complexity. By varying sample time and reward, different agents were created. Analysis is done based on the obstacle avoidance and maximum distance travelled by mobile robot. By varying number of nodes and connection distance feasible paths found using PRM. By integrating these two DDPG and PRM techniques, the mobile robot path length to reach the destination is reduced by 40%.**

Keywords— DDPG, PRM, Reinforcement Learning, Autonomous Navigation, Dynamic Environment

I. INTRODUCTION

Autonomous navigation is the ability of a system, typically a robot or vehicle, to move and operate in an environment without the need for human intervention. This technology has become a cornerstone in a variety of industries, including transportation, robotics, aerospace, and defence, as it enables systems to perform tasks independently, making real-time decisions based on their surroundings [1]. Autonomous navigation is often seen in self-driving cars, drones, robotic vacuum cleaners, and planetary rovers, where these systems must perceive their environment, process the information, and act in a way that ensures safety and efficiency [2].

At the core of autonomous navigation is a combination of sensors, algorithms, and actuators that allow machines to interact with the physical world. The system's ability to autonomously navigate depends on its capability to perform key functions such as perception, localization, path planning, for control and decision making [3].

Autonomous navigation presents several challenges that must be addressed to enable effective and safe operation in real-world environments. Some of the key challenges includes: Obstacle detection and its avoidance, path planning in complex and dynamic environments along with these real-time decision-making is essential, requiring the system to process data and re-plan paths quickly for the dynamic conditions is challengeable [4].

Traditional navigation methods often struggle in dynamic or cluttered spaces, prompting the use of a more adaptive and efficient solution. By focusing on environments of different complexities, the project aimed to create a versatile robot capable of navigating in diverse scenarios [5]. The use of Deep Deterministic Policy Gradient (DDPG) for obstacle avoidance was motivated by the desire to create a system that could learn optimal behaviours through interaction with the environment. DDPG enables the robot to adapt to different challenges it encounters, allowing for better performance in dynamic and unpredictable environments. This adaptability is critical for real-world applications where static solutions may not suffice [6].

To complement the learning-based approach of DDPG, the Probabilistic Roadmap (PRM) was integrated into the system. PRM provides a structural framework by generating a roadmap that guides the robot with a shorter path to the destination. The motivation behind this was to improve the robot's ability to plan collision-free paths more efficiently. When PRM is combined with DDPG, results in shorter and more effective paths.

II. DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

The Deep Deterministic Policy Gradient (DDPG) algorithm is an advanced reinforcement learning (RL) technique designed specifically for handling environments with continuous action spaces. It combines ideas from both Q-learning and policy gradient methods, making it highly effective for complex control tasks, such as those involved in autonomous robot navigation [7]. The agent and environment interaction is shown in fig.1.

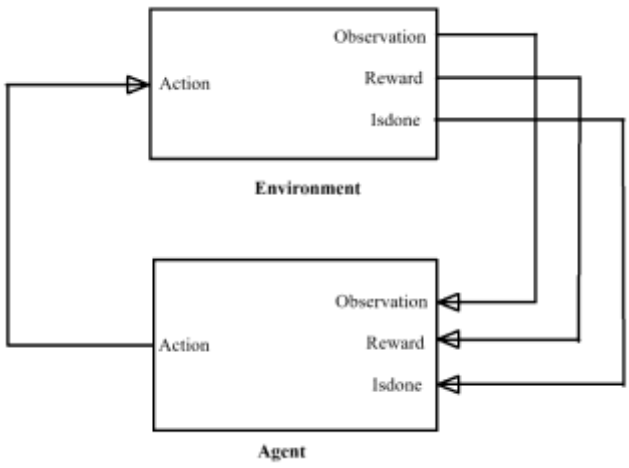

Fig. 1. Agent and Environment interaction in DDPG

The above figure is outline how a DDPG (Deep Deterministic Policy Gradient) agent interacts with the environment in MATLAB during reinforcement learning. Here's a detailed explanation of how DDPG functions in this setup:

A. Environment

The environment represents the simulated system where the agent operates. It could be a robotic system, such as a mobile robot navigating a space. Here, the environment receives the action from the agent (e.g., movement commands or control signals) and responds by updating its state. After processing the action, the environment generates three key outputs: an observation, a reward and is-done flag [8].

B. Agent

The agent is the DDPG-based algorithm that interacts with the environment. The agent has two main components: Actor Network and Critic Network. Actor Network generates continuous actions based on the observations received from the environment. The Critic network evaluates the quality of the actions by estimating the expected cumulative reward (Q-value) [8].

C. Process

The interaction process between the agent and the environment occurs in a loop processes. First, the agent observes the current state (observation). Based on the observation, the agent generates an action using the actor network. Which, the environment applies the action and returns the next observation, reward, and done signal. Meanwhile, the critic network updates the Q-value estimation based on the observed reward. The loop continues until the episode ends (when IsDone is true) [8-9].

III. DDPG Working

The Deep Deterministic Policy Gradient (DDPG) algorithm is an off-policy actor-critic method for environments with continuous action spaces. It enables a DDPG agent to learn a deterministic policy, using a Q-value function critic to estimate the value of the optimal policy. The DDPG features both a target actor and target critic network, along with an experience replay buffer to store past experiences [10]. Additionally, DDPG supports offline training, where agents are trained using saved data without interaction with the environment [11].

In MATLAB's Reinforcement Learning Toolbox, a DDPG agent is implemented using the rlDDPGAgent object. The algorithm supports environments with either continuous or discrete observation spaces, but the action space must be continuous, as descripted in Table-I.

TABLE I. Observation and Action Spaces

Observation Space	Action Space
Continuous or discrete	Continuous

A. Actor and Critic Functions

The DDPG agent relies on two primary functions for training: the Critic and Actor. The Critic is a Q-value function ($Q(S,A)$), created using the `rlQValueFunction` object. The actor is a deterministic policy ($\pi(S)$), created using the `rlContinuousDeterministicActor` object [12].

B. Trainging period

During the training period, the DDPG agent updates the parameters of both the actor and critic at each time step. It stores past experiences in a circular experience buffer. The agent updates the models by sampling mini-batches of experiences from this buffer. The agent perturbs the action chosen by the policy using a stochastic noise model to encourage exploration at every training step. Along-with these rewards are used to evaluate the agent and guiding the agent for to maximize long-term best performance [12].

C. Reward function while training

The agent is rewarded to avoid the nearest obstacle, which minimizes the worst-case scenario. Additionally, the agent is given a positive reward for higher linear speeds, and is given a negative reward for higher angular speeds. This rewarding strategy discourages the agent's behaviour of going in circles. Tuning your rewards is key to properly training an agent, so your rewards vary depending on your application.

The reward function for the agent is modelled as shown in fig. 2.

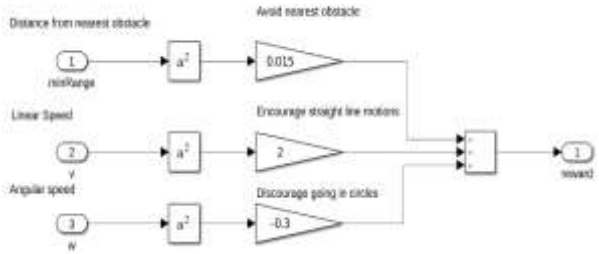

Fig. 2. Reward function for agent

D. Path Planning:

Path planning involves determining a sequence of actions for a robot to follow in order to navigate from a starting point to a goal location while avoiding obstacles and minimizing costs such as time or energy. Various path planning techniques exist, ranging from deterministic algorithms to probabilistic methods. One specific method mentioned is the Probabilistic Roadmap (PRM) method [13-14]. PRM is a sampling-based algorithm that constructs a graph representation of the environment by sampling feasible configurations of the robot and connecting them with collision-free paths between any two points [15].

In this paper, leveraging the RL-based DDPG algorithm to train an autonomous agent to learn navigation policies, and integrate it with the PRM method to generate feasible paths in complex environments. By combining RL and path planning techniques, the project aims to enhance the adaptability and robustness of mobile robots in real-world scenarios, enabling them to navigate efficiently while avoiding obstacles and reaching their destinations faster and safely.

IV. Probabilistic Roadmap (PRM)

Probabilistic Roadmaps (PRMs) are a popular motion planning algorithm used in robotics and computer graphics. They provide a probabilistic approach to finding feasible paths between a starting point and a goal point in a complex environment [16].

PRM Architecture is explained below [17]:

- **Random Sampling:** A large number of random points are sampled within the environment. These points represent potential configurations or poses for the robot.
- **Nearest Neighbour Search:** For each sampled point, its nearest neighbours are identified. This creates a graph-like structure where nodes are the sampled points and edges connect neighbouring points.
- **Edge Validation:** Each edge is checked to ensure it is collision-free. This means that the robot can move from one point to another without colliding with obstacles.
- **Query Processing:** When a query is given (e.g., find a path from point A to point B), the algorithm searches for a path within the graph. This involves finding a

sequence of connected edges that leads from the starting point to the goal point.

- **Smoothing:** The found path can be smoothed to make it more efficient or natural. This involves finding a shorter or more direct path between the same start and goal points.

Several mathematical concepts and formulas are involved in the Probabilistic Roadmap (PRM) algorithm, especially in the processes of sampling, connecting nodes, and searching for a feasible path.

V. METHODOLOGY

In the developed RL based DDPG and PRM model, the robot will reach to the destination through shortest path developed by PRM and safely through DDPG technique.

In the context of the autonomous navigation, an environment for to use RL based DDPG and PRM technique. The developed is shown in the fig.3.

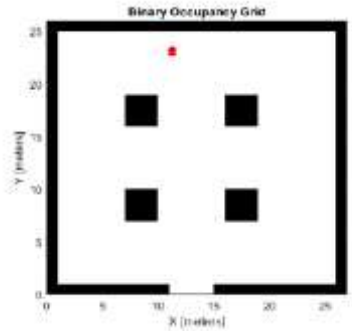

Fig. 3. Environemt used for to simulate RL based DDPG and PRM

In the above environments dark lines represent boundaries and white space is where the robot move. Robot is represented by red dot. The time taken by the robot to reach the destination is minimum and covers the optimal distance.

VI. RL BASED DDPG AND PRM INTEGRATION

This proposes an integration of two techniques i.e.., Deep Deterministic Policy Gradient (DDPG) and Probabilistic Roadmap (PRM) for autonomous navigation of mobile robots. For mobile robots, DDPG enables continuous learning, adaptation to complex environments, and avoiding obstacles [18].

PRM is useful in generating feasible path in complex environments by mapping out multiple routes, reducing computational complexity [19].

For training of the agents, the sample time and reward structure followed for the environment. For integration of the both the techniques the sample time taken was 0.1, reward taken as 400 for DDPG and number of nodes 300-400, connection distance 2,3 in PRM.

A. *Implementation of DDPG algorithm*

The steps followed in training the agent and simulations is as follows.

- **Step 1:** Environment creation using logical matrix.
- **Step 2:** Setting the robot's initial parameters.
- **Step 3:** Training the agent using DDPG algorithm.
- **Step 4:** Simulating the trained agent and tracing the path.

1) Environment setup

The environment created using the logical matrix in MATLAB as shown in fig.4. The logical matrix for the above environment is as follows. In this 1 represents the occupied space and 0 represents the free space where the robot can move.

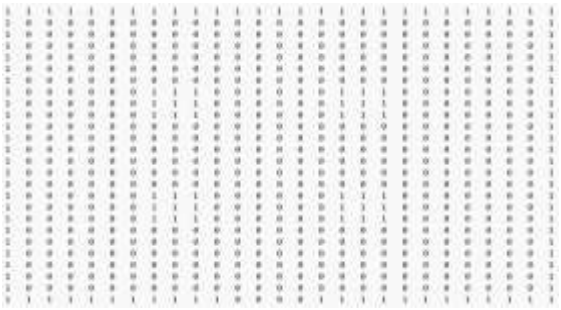

Fig. 4. Logical matrix for environment 2

2) Robot's initial parameters

The red dot represents the robot. After giving the initial robot parameters, the environment looks as shown in the fig.5.

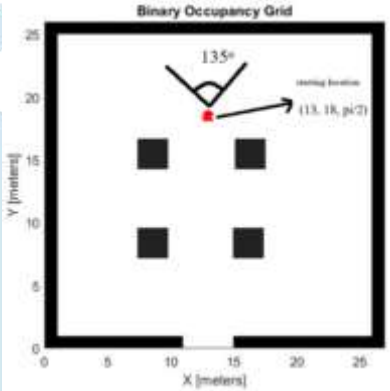

Fig. 5. Robot position with initial parameters

The robot parameters are given as follows:

Range sensor (LIDAR) parameters:

- Scan angles: $-3\pi/8$ to $3\pi/8$ at intervals of $\pi/8$ i.e.., $-3\pi/8$, $-2\pi/8$, $-\pi/8$, 0, $\pi/8$, $2\pi/8$, $3\pi/8$.
- Max Range: The maximum range of the LIDAR sensor (9 units).

Robot parameters:

- Maximum linear speed: 0.3 units/second.
- Maximum angular speed: 0.5 radians/second.
- Initial position and orientation: X=13, Y=18, Theta=$\pi/2$).

With the above things defined are the robot environment in which the robot moves and the robot initial parameters i.e.. the starting settings for the robot.

3) Training parameters:

After setting up the environment and robot parameters the training paraments are set. The introduction part covered the things like actor, agent and reward. Here below only main parameters needed for the training are mentioned.

- Total simulation time: 150 seconds
- Sample time: The time step for the simulation, set to 0.05 seconds.
- Maximum number of episodes for training termination: 10000.
- Maximum steps per episode: Simulation time/ sample time.

- Stop training: When average reward reaches 200.
- Average window length: 30.

After defining all the parameters above, the agent training is done. After reaching the specified cumulative average reward the agent is training is stopped. There are other parameters also defined to stop the training process which includes maximum number of episodes. Each episode is stopped when the robot is hit with the obstacle or reaching the maximum steps defined for episode [20]. After reaching the either of the defined parameters the training is stopped and saved. Reaching the reward means agent is learned to navigate the environment effectively. The training progress of DDPG Agent in a training episode manager of MATLAB is shown in fig.6.

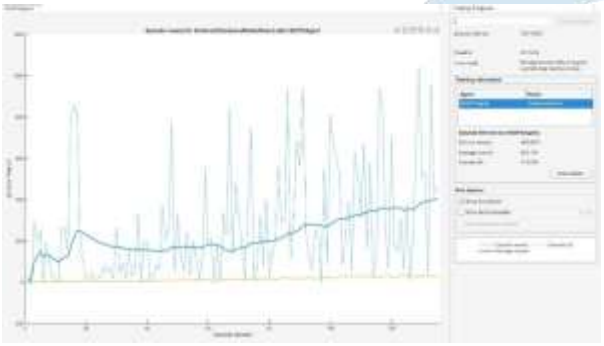

Fig. 6. Training episode manager of the agent.

In which X - axis represents the episode number, Y - axis represents the reward value for that episode. It also contains valuable information like episode number which is current running episode number, duration represents the time it has taken till, episode reward is the reward value that episode, average reward is the ratio of the sum of the reward values so far to the total number of episodes done. Episode Q0 value represents the expected reward for taking a particular decision by the agent. The path traced by the robot in simulation is shown in fig.7. In the environment, the red line represents the path taken by the robot during the simulation.

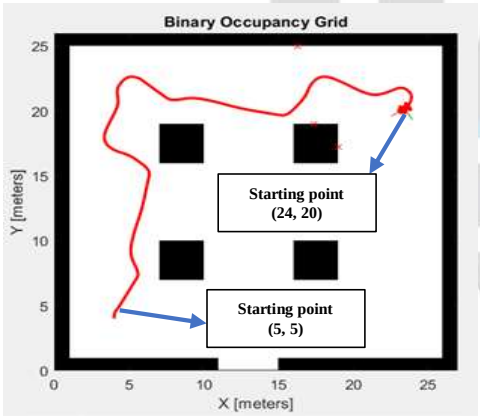

Fig. 7. Path taken by the robot in simulation with DDPG technique

B. Implementation Probabilistic Roadmap (PRM)

The steps followed during the path generation using PRM is as follows.

- **Step1:** Environment setup to convert the environment into binary occupancy grid.
- **Step2:** Inflate the map according to the robot radius.
- **Step3:** Giving the parameters like starting points, ending points, number of nodes, connection distance.
- **Step4:** Generate the path using PRM.

1) Environment setup

In case of PRM, the environment can be of the any type like logical matrix or image captured, first it needs to be converted in to binary occupancy grid. A binary occupancy grid is of two colours one is black and another is white. PRM detects black as obstacle and white as unoccupied space. For the environment, at first its need to be converted into the binary occupancy grid. The created binary occupancy grid of the environment is shown in fig.8.

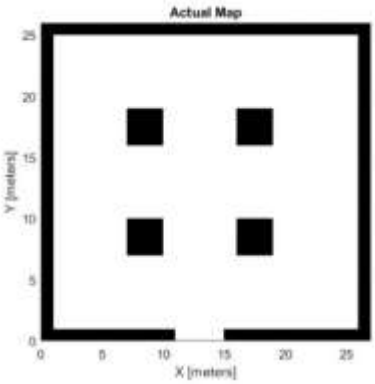

Fig. 8. Environment- binary occupancy grid.

Black colour: Represents occupied space.

White colour: Represents un occupied space.

2) Inflating Map:

Now the next step in finding the path using PRM is to inflate the map. Inflation of the map is done for taking account of the robot radius. So that the robot will not touch the edges and corners of the environment.

- Robot radius: 0.02 units

In the above case, with mentioned robot radius, the free space cell next to each of the occupied cell is converted to occupied cell. It is applied to each side of the occupied cell as shown in Fig.9.

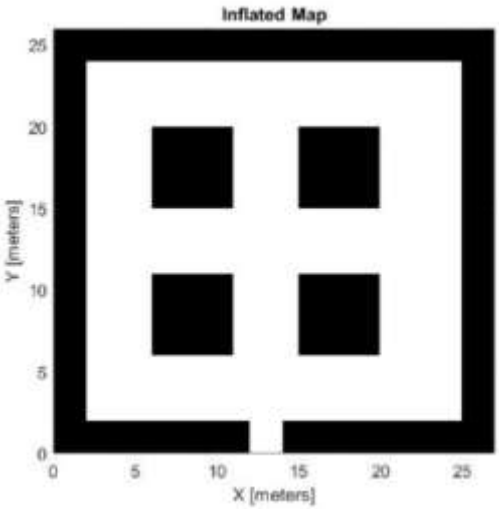

Fig. 9. Inflated Environment-inflated map.

3) Induced Nodes Map

Next step in finding the path using PRM is to define number of nodes. Nodes acts as junction block for path in PRM. The path is generated by connecting the nodes [21]. In the fig.10, the blue dots represent the node points. In the number of nodes taken in the shown fig.10 is 350 nodes.

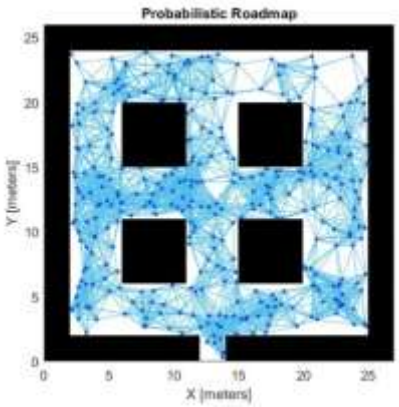

Fig. 10. Environment - number of nodes defined

These values can be varied depending on the size of the environment.

- Number of nodes: 350
- Now the starting and ending locations are set.
- Starting location: (5, 5)
- Ending location: (23, 23)
- Now, the connection distance (Maximum Connection distance between nodes) is defined
- Connection distance: 1, 2, 3, 4 and 5

4) Path generated using PRM

The path is generated using PRM technique is shown as the red line in the fig.11.

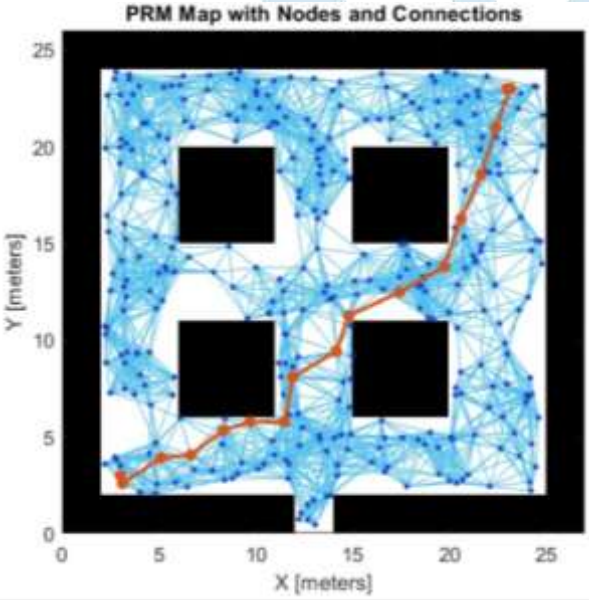

Fig. 11. Environment- path generated using PRM

5) Path generated by PRM with different connection distance

The following fig.12 is the path generated using PRM, with 350 nodes and connection distances of 2. The fig.13 is the path generated using PRM, with 350 nodes and connection distances of 3. The following fig.14 is the path generated using PRM, with 350 nodes and connection distances of 4. The fig.15 is the path generated using PRM, with 350 nodes and connection distances of 5.

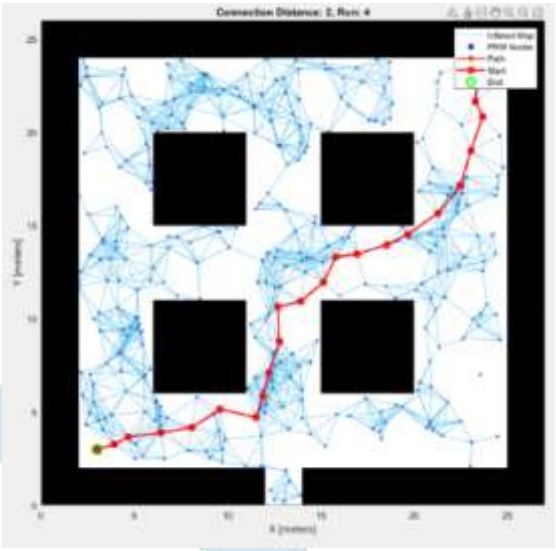

Fig. 12. PRM with connection distance 2 for 350 nodes

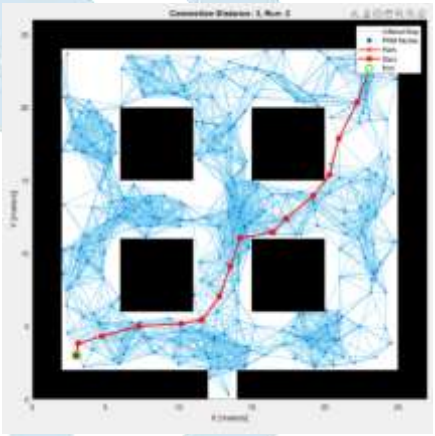

Fig. 13. PRM with connection distance 3 for 350 nodes

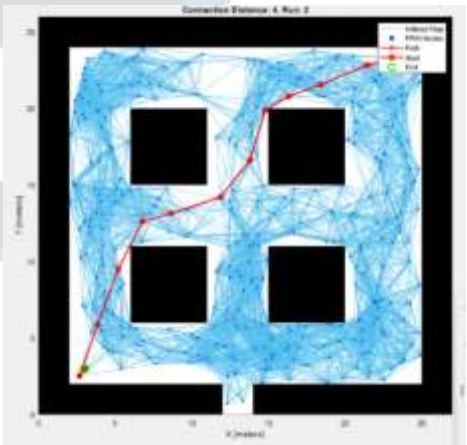

Fig. 14. PRM with connection distance 4 for 350 nodes

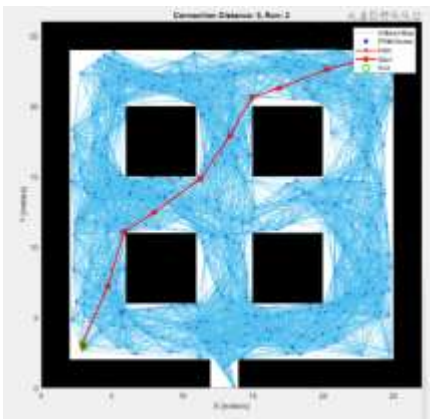

Fig. 15. PRM with Connection distance 5 for 350 nodes

The above are the sample results of the PRM in environment. Like this the testing is done with number of nodes 50, 70, 100, 125, 150, 200, 225, 250, 300, 350, 400, 450, 500, 550 and 600, with connection distances 1, 2, 3, 4 and 5. Each test is done ten times and success rate, simulation time, length of the path etc are found for environment. The results are tabulated in results section. From this optimal range of number of nodes for the environment are found, and same applied for any environments. For other environment same procedure is followed with the selected range of number of nodes and with varying connection distance results are obtained and tabulated in Table-II. By analysing the results optimal conditions for the PRM is found.

TABLE II. PATH DISTANCE WITH VARIATION OF CONNECTION DISTANCE

Location Points	No. of Nodes	Connection distance	Average No. of. nodes	Average path distance
Starting Point-(3,3) Ending Point-(23,23)	350	1	0	0
		2	25.6	34.616
		3	17.8	32.099
		4	14.3	32.036
		5	12.2	31.369

C. Integration of PRM and DDPG

The procedure followed for integration of DDPG and PRM is as follows.

- **Step 1:** Using PRM optimal conditions find the feasible path between two points.
- **Step 2:** Node points generated in the feasible path using PRM, the location coordinates of node points are extracted.
- **Step 3:** The extracted node points are given as input to the DDPG code, so that after reaching each node, the orientation is updated for the mobile robot in the direction towards the next node. This is achieved by finding the slope between two nodes every time. The slope formula used here is $((y_2-y_1)/(x_2-x_1))$. Where (x_1, y_1) represent current coordinates, (x_2, y_2) represent next coordinates.

The following are the node points in the path generated in all the environments using PRM. The number of nodes given during PRM are 350 and connection distance as 3.

For the environment, results are obtained for the mobile robot as the starting location is (23, 20) and ending location is (6, 4). The Fig.16 shows the path generated by only DDPG The Fig.17 shows the path generated by the PRM. The Fig.18 shows the path traced during using of PRM and DDPG algorithms.

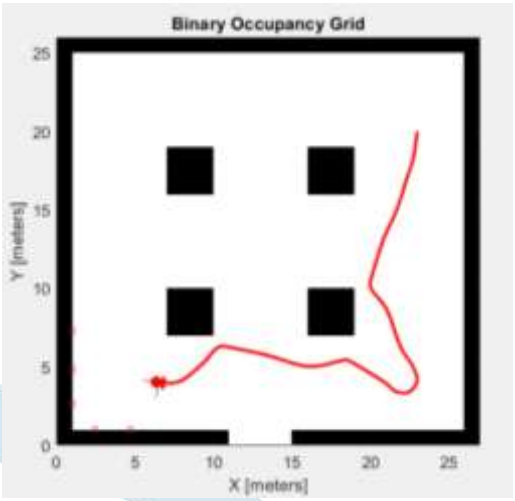

Fig. 16. Path followed by mobile robot using DDPG

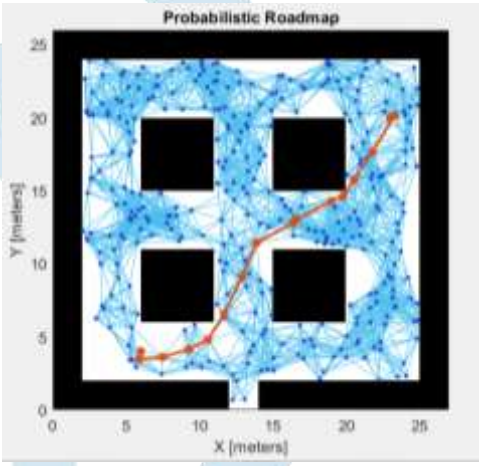

Fig. 17. Path generated by PRM. (path length 26.53 units)

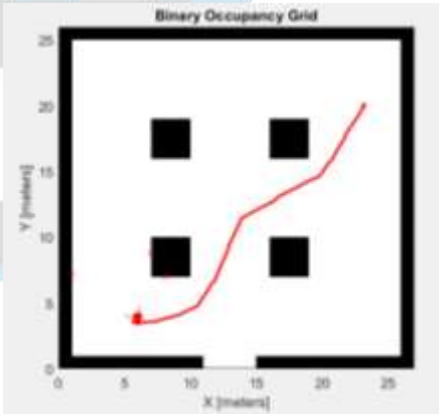

Fig. 18. Mobile robot path traced by integrating PRM+DDPG

VII. CONCLUSION

The mobile robot successfully reached the destination from a given starting point while avoiding obstacles, during the integration of PRM and DDPG, with the selected sample time of 0.1 and reward of 400 points. Using 350 nodes and a connection distance of 3, the algorithm effectively found a path between the selected points. On average, this integration enabled the mobile robot to save 40% in both distance and time.

In this paper, Reinforcement Learning-based Deep Deterministic Policy Gradient (DDPG) is integrated with the Probabilistic Roadmap (PRM) technique to enhance mobile robot navigation within defined environments. This integration reduced the path length by over 40% compared to using DDPG alone, offering a more efficient approach to navigation.

REFERENCES

[1] R. Van Hoa, T. D. Chuyen, N. T. Lam, T. N. Son, N. D. Dien and V. T. T. Linh, "Reinforcement Learning based Method for Autonomous Navigation of Mobile Robots in Unknown Environments," 2020 International Conference on Advanced Mechatronic Systems (ICAMechS), Hanoi, Vietnam, 2020, pp. 266-269, doi:10.1109/ICAMechS49982.2020.9310129.

[2] Luo, Qian, et al. "Localization and navigation in autonomous driving: Threats and countermeasures." *IEEE Wireless Communications* 26.4 (2019): 38-45..

[3] Ribeiro, T., Gonçalves, F., Garcia, I., Lopes, G., & Ribeiro, A. F. 2019. "Q-Learning for Autonomous Mobile Robot Obstacle Avoidance." In *Proceedings of the 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 1-7. Porto, Portugal. doi:10.1109/ICARSC.2019.8733621.

[4] Kiumarsi, B., Vamvoudakis, K. G., Modares, H., & Lewis, F. L. 2018. "Optimal and Autonomous Control Using Reinforcement Learning: A Survey." *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2042-2062. doi:10.1109/TNNLS.2017.2773458.

[5] Wang, B., Liu, Z., Li, Q., & Prorok, A. 2020. "Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning." *IEEE Robotics and Automation Letters*, 5(4):6932-6939. doi:10.1109/LRA.2020.3026638.

[6] P. Phueakthong, J. Varagul and N. Pinrath, "Deep Reinforcement Learning Based Mobile Robot Navigation in Unknown Environment with Continuous Action Space," 2022 5th International Conference on Intelligent Autonomous Systems (ICoIAS), Dalian, China, 2022, pp. 154-158, doi: 10.1109/ICoIAS56028.2022.9931272.

[7] H. Tan, "Reinforcement Learning with Deep Deterministic Policy Gradient," 2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA), Xi'an, China, 2021, pp. 82-85, doi: 10.1109/CAIBDA53561.2021.00025.

[8] Mao, Hangyu, et al. "Modelling the dynamic joint policy of teammates with attention multi-agent DDPG." arXiv preprint arXiv:1811.07029 (2018).

[9] Ma, Jinlin, et al. "A decision-making of autonomous driving method based on DDPG with pretraining." *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* (2024): 09544070241227303

[10] Huang, Wei, et al. "Parametric dueling DQN-and DDPG-based approach for optimal operation of microgrids." *Processes* 12.9 (2024): 1822.

[11] Fujimoto, Scott, David Meger, and Doina Precup. "Off-policy deep reinforcement learning without exploration." *International conference on machine learning*. PMLR, 2019.

[12] Lin Li, Yuze Li, Wei Wei, Yujia Zhang, Jiye Liang, Multi-actor mechanism for actor-critic reinforcement learning, Information Sciences, Volume 647, 2023, 119494, ISSN 0020-0255.

[13] Deshpande, Shripad V., et al. "Mobile robot path planning using deep deterministic policy gradient with differential gaming (DDPG-DG) exploration." *Cognitive Robotics* 4 (2024): 156-173.

[14] Hu, Wenjie, et al. "Double Critics and Double Actors Deep Deterministic Policy Gradient for Mobile Robot Navigation using Adaptive Parameter Space Noise and Parallel Experience Replay." *IEEE Access* (2024).

[15] Karaman, Mustafa, and Emilio Frazzoli. 2011. "Sampling-based Algorithms for Optimal Motion Planning." International Journal of Robotics Research 30(7):846-894.

[16] Bohlin, Robert, and Lydia E. Kavraki. "Path planning using lazy PRM." *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE, 2000.

[17] Kavraki, L. E., Švestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4), 566–580.

[18] Bai, Xueqian, and Haonian Wang. "An improved DDPG algorithm based on evolution-guided transfer in reinforcement learning." Journal of Physics: Conference Series. Vol. 2711. No. 1. IOP Publishing, 2024.

[19] Li, Q.; Xu, Y.; Bu, S.; Yang, J. Smart Vehicle Path Planning Based on Modified PRM Algorithm. Sensors 2022, 22, 6581. https://doi.org/10.3390/s22176581.

[20] Dankwa, Stephen, and Wenfeng Zheng. "Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent." *Proceedings of the 3rd international conference on vision, image and signal processing*. 2019.

[21] Mohamed Reda, Ahmed Onsy, Amira Y. Haikal, Ali Ghanbari, Path planning algorithms in the autonomous driving system: A comprehensive review, Robotics and Autonomous Systems, Volume 174, 2024, 104630, ISSN 0921-8890.