ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 7, Issue 3, March 2020

Reinforcement Learning based path planning controller for collision avoidance and goal seeking of a mobile robot in complex dynamic environment

[1] Arun Shankar M, [2] Dr P S Lalpriya
[1] Dept. Interdisciplinary Research, Electronics and Communication, College of Engineering Trivandrum, Kerala
[2] Professor, Dept. Electrical Engineering, College of Engineering Trivandrum, Trivandrum

Abstract: A path planning controller of a mobile robot is very critical considering the fact that the robot navigation should happen reaching the target without hitting the obstacles. But the complexity of the task drastically varies with the type of environment through which the robot navigates. In certain environments like airports, shopping complex, bus terminals etc. the environment is so dynamic such that the robot navigation done by different path planning approaches which works well in static environments won't be suitable for these dynamic applications. In this paper a path planning controller for collision avoidance and goal seeking of a mobile robot is presented utilizing deep Q reinforcement learning algorithm. A dynamic environment is created using Robot operating system Gazebo simulator and one of the most popular open source robot Turtlebot3 is used for simulation. The hyper-parameters are selected in such a way that the reinforcement learning path planner will train the robot to form a policy which will maximize the reward function. A hardware model also developed utilizing ultrasonic sensors for obstacle avoidance and UWB technology for localization and goal tracking. Similar results are obtained from the hardware model also when it is trained using the path planner.

Keywords—Reinforcement Learning; Mobile Robot; Dynamic Environment ; styling; insert

INTRODUCTION

Over the course of time several path planning algorithms were proposed for robots in dynamic environments. Some of them are based on algorithms such as Bug algorithm ,potential fields algorithm and A* algorithm with multiresolution grids [1].Even though these algorithms can be used for path planning applications to an extent these methods inherently possess many drawbacks such as Bug algorithm or its modified versions are based on the binary methods, also in order to avoid collisions they tends to make larger radius than the usual. Bug algorithms always follow direct path towards the goal and on the way whenever they are encountered with obstacles, they tend to turn around the obstacles in the same direction. One of its modified versions uses the technique to follow a tangential path towards the nearest obstacle and follow the standard bug algorithm as usual. For finding the side of the tangential path with respect to the obstacle the tangent which forms shortest area is selected. Similarly A* algorithms which is based on the concept of representing environment as a grid and is divided in to nodes or pixels, the approach is to find the optimal path between the nodes sounds good on theory but the computational requirement is so high as the number of cells in the grid increasing or resolution of images used to represent the environment is very less leading to more processing time and collision. In potential field algorithm the robot moves similar to the case of other force fields such that it directed towards the goal and is repelled in the vicinity of obstacles. The potential field forces must be set in order to find the optimal path is one of the main disadvantages of this method. Several deterministic and probabilistic algorithms are also available for path planning [2-5]. Dijkstra, Visibility Graph, PRM, L-PRM, RRT, and B-RRT are some examples of these algorithms. Which can be used for very specific applications only.

In Reinforcement learning according to the environment the learner will take actions in such a way that to maximize the reward function. There are several reinforcement learning algorithms are proposed in dynamic environment [6]. Some improvements are proposed in traditional RL to make robotic navigation more efficient such as implementation of a forgetting mechanism, hierarchical structuring of reinforcement learning agent. Another method for robotic navigation is also proposed which is purely based on sensor data

ISSN (Online) 2394-2320

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 7, Issue 3, March 2020

instead of relying on spatial mapping information [7]. From the dynamics of the sensor data the robot will form a reward policy which will find the places where the robot may navigate in future. Since the robot is relaying only on sensor data the path formed may not be accurate and it will not lead to goal tracking also. In another method two modules are trained separately and combined together using a switching function which will activate one module at a time one for obstacle avoidance and another for goal tracking [8]

Some significant works are also available with respect to q learning and deep q-learning [9-12], which is the reinforcement learning method used in this work. In order to improve the q learning method a multi q-table for q-learning is suggested [9] in some of the works which will creates a new q table whenever a sub goal is obtained. For avoiding overestimation of discounted rewards concepts of approximation spaces also included in some of the improvements proposed [10]

For localization and mapping of the environment there are several methods are available particularly SLAM is the prevalent one. In one of the comparative studies conducted between SLAM and UWB technology [13] there are several advantages are there with UWB technology due to its high bandwidth and its Time of Arrival method than recursively calculating pose and orientation done by SLAM.

In this paper we focused on Deep Q reinforcement learning algorithm for dynamic obstacle avoidance and goal tracking. A dynamic environment is created using Gazebo simulator of Robot operating system and one of the most popular open source robot TurtleBot3 is used for simulation in the created environment. Section II of the paper explains basic theory behind the paper which explained in separate subheadings and in Section III involves explanation of simulation model in Gazebo environment. Section IV involves how the experiment setup recreated with fabrication model and Section V explains results and observations obtained from these experiments. Section VI includes conclusions and future scope of this paper.

II. BASIC THEORY

A. Reinforcement Learning

In reinforcement learning the learner will discover the proper actions corresponding to various situations in order to maximize the reward function so that the learner is learning by itself what to do and what not to do. So, the actions are not only affecting the immediate rewards but also it will affect next states and the upcoming rewards. Reinforcement learning is very much different from supervised learning and unsupervised learning categories of machine learning, because in supervised learning there is a knowledgeable supervisor is required for labelling the training set and unsupervised learning is finding the hidden structure of the unlabeled data. Even though like unsupervised learning it does not require a knowledgeable supervisor it differs because of the fact that reinforcement learning trying to maximize the reward function instead of finding the hidden structure in the data. Along with agent and environment there are other four important terms associated with RL they are policy, reward, value function and model. Policies are the core of RL agent because it maps what actions to be taken in each state. In each iteration or each step, the environment responds back to the system with a number called the reward. Over the course of training the agent will try to maximize its reward received from the environment. Value function defines how much reward the agent can accumulate in the long run starting from the initialization. If we require planning to solve RL related problems then the method is known as model-based methods and instead if it uses trial and error scenarios then that method is known as model free method. There are several algorithms are available for RL and in this work Deep Q learning which is coming under the category of Temporal Difference learning has been used.

B. Temporal Difference Learning

Temporal difference learning is a combination of two of the most important ideas or methods used in RL such as Monte Carlo methods (MC) and Dynamic programming (DP). Similar to Monte Carlo methods TD learning can also estimate the policy without the dynamic model of the environment and similar to Dynamic programming without getting the final outcome they update their estimates from the previous estimates they already learned. There are advantages of TD learning over the other two and that's why TD algorithms are very much suitable for mobile robot application. In MC methods agent has to wait for an episode to end in order to get the returns but in TD only one-time step wait is required. Because it will be an important consideration

ISSN (Online) 2394-2320

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 7, Issue 3, March 2020

when some episodes take more time to finish. When comparing with DP it does not require a model of the environment to learn. These combined advantages make TD method is very much appropriate for path planning.

C. Q- Learning

Q learning is one of the off-policy TD algorithms in which the action-value function directly approximates the optimal action-value function, independent of the policy being followed (Watkins, 1989). Hence it simplifies the algorithm and converges faster. The policy determines which are the state-action pairs being updated. Q-learning has several variants and if it uses any convolutional neural network then it is known as deep q neural network. But when we use nonlinear functions to approximate Q value, it may not be stable and not very easy to converge. But when we use an experience replay technique then instead of looking for most recent actions to proceed it uses random sample of prior actions so it avoids correlations in the observation sequence and smoothens the changes in our data distribution. It is first patented by Google DeepMind to play Atari games with nearly super human power.

D. Robot Operating System

Robot operating system is a very important framework for various robotic applications. It consists of a number of inter connected nodes that passes messages in between. ROS consists of a number of tools such as MoveIt, Gazebo etc. and a number of libraries that simplifies complex robotic tasks. ROS software modules can be written in any programming languages such as C, CPP, Python, and Java are some among them. ROS consists of several nodes (individual programs) and edges (message streams) for communicating between these nodes. Roscore is the service that facilitates the connection between the nodes. There is a publisher-subscriber model between the nodes and each node connects to roscore to provide details about the message streams they want to pass in this publisher-subscriber peer to peer connection. There are thousands of ROS packages available as open source (a combination of codes, data and its documentation) makes the usage of ROS very effectively.

E. Gazebo Simulator

Gazebo is one of the most widely used robotic simulator by the robotics community because of its open source availability and compatible with ROS. Gazebo can be used for Dynamic simulation. Initially it supported only Open Dynamics engine but in newer versions it supports various physical engines (Bullet, Simbody, DART etc.). For 3D graphics it uses open source graphics rendering engine which supports light, shadow etc. Gazebo simulator supports various sensors and plugins. Very commonly used Laser scanner, 2D/3D camera, Depth camera, various contact sensors all can be integrated with gazebo. Plug-ins can be created or commonly used plug-ins are supported in gazebo. Other supported features include TCP/IP data transmission, Cloud simulation and command line tool supports both GUI and CUI.

F. Turtlebot

Turtlebot is a widely used mobile robot for education and research purpose. It is cheaper and easy to build and comes with open source software. Various Turtlebot versions have been deployed over the years namely Turtlebot1, Turtlebot2 and Turtlebot3. Their plugins are commonly used for simulation purpose. Turtlebot consists of a robot base, distance sensor and associated controls. Turtlebot3 is an advanced version over its predecessors by having a Dynamixel actuator. It is a single module consists of controller, sensor, motor etc. all are integrated together.

III. MODEL SIMULATION IN GAZEBO ENVIRONMENT

For simulating the robot motion in static and dynamic environment in gazebo simulator suitable worlds are created. A world with only walls, partition and corners used for static obstacle avoidance. A world with moving cylinders and static walls is used for dynamic obstacle avoidance. While creating the world pose, dimensions, Material and collision details of each of the cylinders and other components are given. Physics type and velocity constrains are provided for the cylinders such that its motion is defined in the environment. Launch files are created for launching the world in gazebo.

Setting the parameters are the first step towards deep Q reinforcement learning. The parameter used in the DQN networks are given as in the table given below

ISSN (Online) 2394-2320

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 7, Issue 3, March 2020

TABLE 1

Sl No	*Hyperparameters*	*Value*
1	Time step of one episode	6000
2	Update rate of network	2000
3	Discount factor	0.99
4	Learning rate	0.00025
5	Epsilon	1.0
6	Epsilon decay	0.99
7	Minimum of epsilon	0.05
8	Sample batch size	64
9	Train start size	64
10	Replay memory	1000000

Turtlebot3 is available with different models such as burger, waffle etc. Xacro files of any of the turtlebot3 configuration can be used for training. Files required for turtlebot navigation should install prior to loading of the environment. For training the neural network Tensorflow and Keras are used. In deep Q reinforcement learning the agent is trained in such a way that goal position should be reached after avoiding obstacles. Whenever it reaches its goal position a big positive reward is provided and when it hits an obstacle a big negative reward is provided. The episode ends when the agent reaches the goal or hits the obstacle or after the maximum time limit given to each episode.

After setting the hyperparameters states of the reinforcement agent has been set. State is the current position of the mobile robot in the environment. Visualization of laser data (LDS) and localization gives the current state of the robot. State size of turtlebot can be changed to suitable value from the default state size. Action definition is very much critical in deep Q learning. There are five actions corresponds to five angular velocities of the motor for straight motion, turns with smaller and larger angular velocity in both the direction. Set reward function is very important in any reinforcement learning. Here for collision the reward given as -150 and for reaching the goal the reward given as + 200.

A. Hyperparameters

• Episode step: It is the time step required for one episode

• Update rate: It is the update rate of target network

• Discount factor: It represents the reduction in value of each events according to how far they are from the current value.

• Learning rate: It is the rate with which the learning takes place.

• Epsilon: It is the probability of choosing a random action

• Epsilon decay: It is the rate at which the epsilon decays as the episodes progress

• Epsilon minimum: It is the minimum of epsilon value

• Sample Batch size: It is the size of the group containing training samples.

• Train start size: It is the minimum memory size required to start the training.

• Replay memory: It is the size of replay memory

B. Environments used for training

During the training stage the static environment (fig. 1) made with many curves and turns which makes it hard to learn in the environment. Turtlebot2 is also tried in static environment for a comparison purpose. Dynamic environment is a combination of static and dynamic obstacles hence it is also hard to learn for the agent and makes the training more efficient.

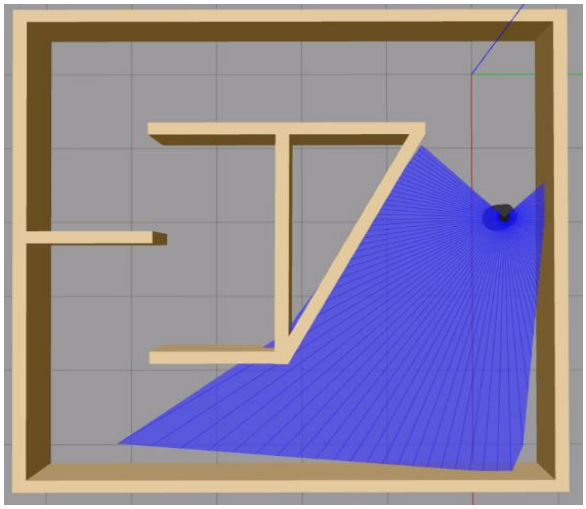

Fig. 1. Static environment

ISSN (Online) 2394-2320

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 7, Issue 3, March 2020

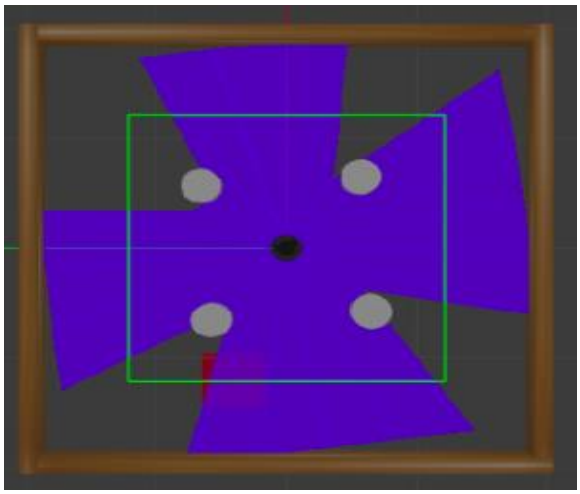

Fig. 2. Dynamic environment

IV. HARDWARE MODEL

For testing the simulation results in real world, a hardware model has been created. For reducing the cost and at the same time in order to meet the strength requirements robot body has been made using the PMMA plastic. Laser cutter is used to cut the PMMA plastic sheet in to required dimension. Differential driven mobile robot is made using two of the SPG30E-20K DC Geared Motor with encoder and a caster wheel is used for support. L293D motor driver is used for driving both the motors. Five ultrasonic sensors (HC-SR04) are used for finding the distance with the obstacles. Ultrasonic sensors are used for getting the distance information because it is cheap and easy to integrate with ROS. DWM 1000 modules are used for real time localization system. Three anchors and one tag are used for finding the real time position. Time taken by the UWB signal to pass between anchor and tag is calculated using the time of flight (TOF) method. Rasberry pi B3 with ubuntu mate 18.04

OS with ROS Melodic installed is used for sending the data between mobile robot and laptop in real time. Ubuntu 18.04 with ROS melodic installed in the laptop and Rasberry pi communicates each other using the client server model. Motor controls are the actions provided as the outputs of the reinforcement algorithm. After successful completion of training hardware model also gave similar results that are obtained from the simulation model.

V. RESULTS AND OBSERVATIONS

After training the robot in the dynamic environment for more than 3000 episodes the robot is able to track the goal while successfully avoiding dynamic obstacles. For an unknown goal position once the robot reaches the goal x-y co-ordinates of the point, number of episodes are displayed in terminal (fig. 3). Action state with current reward and total reward also displayed (fig. 4).

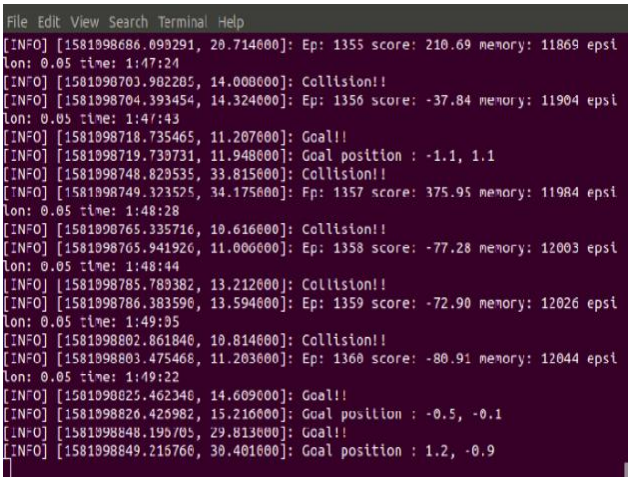

Fig. 3. Terminal display

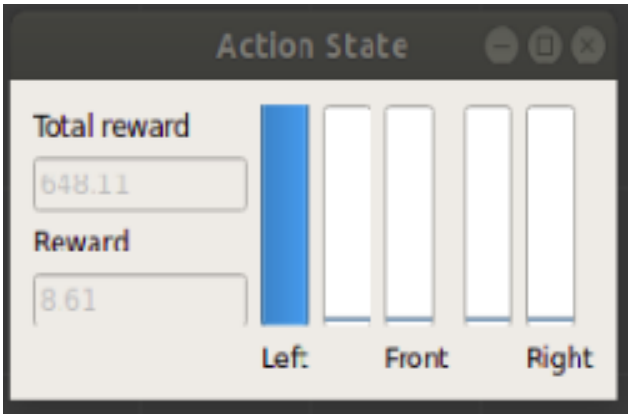

Fig. 4. Action state

A graph is plotted between total reward/Avg maximum Q value Vs number of episodes (fig. 5)

ISSN (Online) 2394-2320

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol 7, Issue 3, March 2020

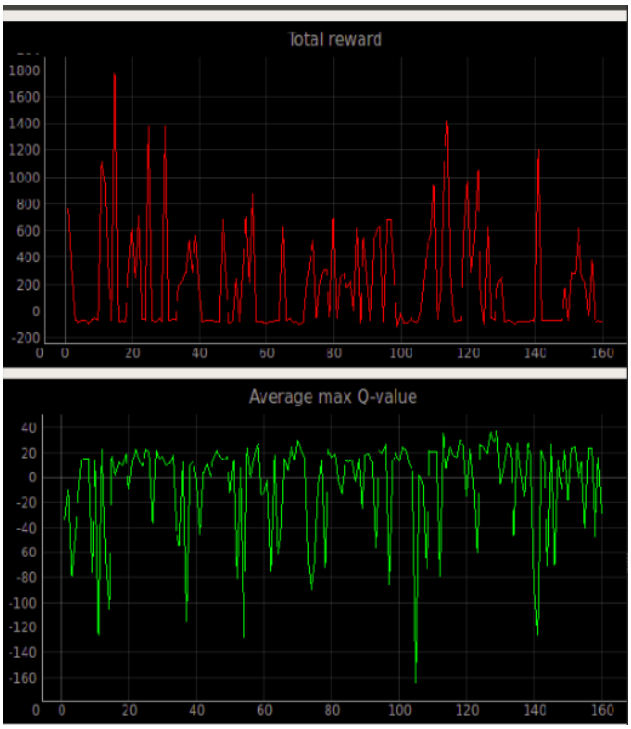

Fig. 5. Total reward/Avg max Q value Vs Episodes graph

VI. CONCLUSIONS AND FUTURE SCOPE

The reinforcement module which is trained in the dynamic environment can be used in other complex environments like museums, malls, Airports etc. As a future work multi agents can be tried with the same algorithm and more advanced or different existing reinforcement algorithms can be tried for fulfilling the same objectives achieved in this paper.

VII. ACKNOWLEDGMENT

This work cannot be completed without the support and encouragement of professors and students of Robotics and automation batch of Dept. of Interdisciplinary Research. I would like to express my special thanks of gratitude towards everyone in the Dept of Interdisciplinary Research.

REFERENCES

[1] F. Haro and M. Torres, ‘‘A comparison of path planning algorithms for omni-directional robots in dynamic environments,’’ in Proc. IEEE 3rd Latin Amer. Robot. Symp., Oct. 2006, pp. 18–25.

[2] A. Alexopoulos, A. Kandil, P. Orzechowski, and E. Badreddin, ‘‘A comparative study of collision avoidance techniques for unmanned aerial vehicles,’’ in Proc. IEEE Int. Conf. Syst., Man, Cybern., Oct. 2013, pp. 1969–1974.

[3] A. Mohammadi, M. Rahimi, and A. A. Suratgar, ‘‘A new path planning and obstacle avoidance algorithm in dynamic environment,’’ in Proc. 22nd Iranian Conf. Elect. Eng. (ICEE), May 2014, pp. 1301–1306.

[4] I. Susnea, V. Minzu, and G. Vasiliu, ‘‘Simple, real-time obstacle avoidancealgorithmformobilerobots,’’inProc.8thWSEASI nt.Conf.Comput. Intell., Man-Mach. Syst. Cybern. (CIMMACS), Stevens Point, WI, USA, 2009, pp. 24–29.

[5] Z.-Q. Ma and Z.-R. Yuan, ‘‘Real-time navigation and obstacle avoidance based on grids method for fast mobile robots,’’ Eng. Appl. Artif. Intell., vol. 8, no. 1, pp. 91–95, 1995.

[6] G. Yen and T. Hickey, ‘‘Reinforcement learning algorithms for robotic navigation in dynamic environments,’’ in Proc. Int. Joint Conf. Neural Netw. (IJCNN), vol. 2, May 2002, pp. 1444–1449.

[7] X.-T. Truong, H. T. Dinh, and C. D. Nguyen, ‘‘An efficient navigation frameworkforautonomousmobilerobotsindynamicenviron mentsusing learningalgorithms,’’J.Comput.Sci.Cybern.,vol.33,no.2,pp .107–118, 2018.

[8] Y. Wang, H. He, and C. Sun, ‘‘Learning to navigate through complex dynamic environment with modular deep reinforcement learning,’’ IEEE Trans. Games, to be published.

[9] V. F. da Silva and A. H. R. Costa, ‘‘Compulsory flow Q-learning: An RL algorithmforrobotnavigationbasedonpartial-policyandmacro-states,’’ J. Brazilian Comput. Soc., vol. 15, pp. 65–75, Sep. 2009.

[10] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, “Continuous Deep QLearning with Model-based Acceleration,” in Proceedings of The 33rd International Conference on Machine Learning, 2016, pp. 2829–2838.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.

[12] A. Ng, A. Coates, M. Diel, V. Ganapathi, J.

Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," Experimental Robotics IX, pp. 363–372, 2006.

[13] M.J.Segura,F.A.A.Cheein,J.M.Toibero,V.Mut,and R.Carelli,''Ultra wide-bandlocalizationandSLAM:Acomparativestudyformobilerobot navigation,'' Sensors, vol. 11, no. 2, pp. 2035–2055, 2011.

[14] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe, ''A review of recentdevelopmentsinsimultaneouslocalizationandmapping,''inProc. 6th Int. Conf. Ind. Inf. Syst., Aug. 2011, pp. 477–482.

[15] S. Kumar, V. Dhiman, M. R. Ganesh, and J. J. Corso, ''Spatiotemporal articulated models for dynamic SLAM,'' CoRR, vol. abs/1604.03526, pp. 1–10, Apr. 2016.

[16] M. Henein, G. Kennedy, V. Ila, and R. Mahony, ''Simultaneous localization and mapping with dynamic rigid objects,'' CoRR, vol. abs/1805.03800, pp. 1–10, May 2018.

[17] I. Randria, M. M. B. Khelifa, M. Bouchouicha, and P. Abellard, ''A comparative study of six basic approaches for path planning towards an autonomous navigation,'' in Proc. 33rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON), Taipei, Taiwan, Nov. 2007, pp. 2730–2735.

[18] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, and H. Youssef, ''Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments,'' Int. J. Adv. Robot. Syst., vol. 14, no. 2, pp. 1–15, 2017.