Proceedings of the 21st International Conference on Modelling and Applied Simulation (MAS), 027
19th International Multidisciplinary Modeling & Simulation Multiconference

2724-0045 © 2022 The Authors.
doi: doi.org/10.46354/i3m.2022.mas.027

Path Planning and Control for Omni-directional Mobile Robot using Q-Learning and CLIK Algorithm for Home Environment

Saurabh Sachan[1,*] and P. M. Pathak[1]

[1]Department of Mechanical and Industrial Engineering, Indian Institute of Technology Roorkee, Roorkee, 247667, India

[*]Corresponding author. Email address: ssachan@me.iitr.ac.in

Abstract

Localization, mapping, and planning are the three crucial steps to accomplishing the autonomous navigation of mobile robots in an unfamiliar environment. Since implementation of reinforcement learning (RL) algorithms for autonomous navigation in the case of omni-directional robots is a less explored research area, and also such robots have a unique feature over differential drive robots that they can also produce sideways movement. Therefore, in this paper, an RL algorithm called Q-learning is used to get the safe and shortest path from a start point (SP) to a goal point (GP) in a home environment. The path trajectories are obtained by using polynomial curve fitting. The closed-loop inverse kinematics (CLIK) algorithm is used to control a three-wheeled omni-directional mobile robot to follow the desired path. The simulation and plotting are done using MATLAB. The simulation results show that the suggested algorithm can effectively recognize and avoid static obstacles of different shapes and dimensions in an indoor home environment.

Keywords: Omni-wheeled robot; Q-learning; CLIK algorithm

1. INTRODUCTION

Autonomous navigation is one of the emerging research area due to availability of different machine learning (ML) algorithms and high computation power. Such navigation requires mobile robots to follow a safe trajectory within unknown environments containing both static and dynamic obstacles. Over past few years, various ML algorithms have been developed for autonomous navigation for different types of obstacles in an environment (Duguleana and Mogan (2016); Syed et al. (2014); Chen et al. (2019b)). For environments with few dynamic obstacles an algorithm based on reinforcement learning (RL) algorithm called Q-learning and neural network was proposed for the solution of path planning problem (Duguleana and Mogan (2016)). The robot had the global knowledge of the environment during navigation. Another study proposed an energy efficient trajectory planning method based on a ML algorithm for industrial robots (Yin et al. (2019)). The three main parts of the research work were data collection, modelling and optimisation process. Sometimes, active simultaneous localization and mapping (SLAM) and deep double Q-network (DDQN) algorithm are also combined to build the environment map and avoid obstacles for robot's navigation (Wen et al. (2020)). In case of omni-directional robot for autonomous navigation, a deep Q-learning network based approach is used which uses Stochastic gradient descent method to update the training data (Manh et al. (2020)). A neural network and hierarchical reinforcement learning (HRL) based model is also created for mobile robots path planning (Yu et al. (2020)). There were some drawbacks to Q-learning algorithm as it converged slowly for the optimized solution. A solution to the problem was given in the research work by (Low

© 2022 The Authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/).

et al. (2019)) where flower pollination algorithm (FPA) was used to initialize the Q-table prior to the implementation of Q-learning, giving faster optimal solutions. (Wang et al. (2020)) also resolved the issue of slow convergence rate during the training part of DDQN. A tree-DDQN (TDDQN) was planned for dynamic path planning of wheeled mobile robots where the tree structure was optimized and the algorithm was able to avoid incomplete and over-detected paths. At the end the best path was selected by using non-maximum suppression method. Another way to enhance the Q-learning algorithm was adopted by (Jaradat et al. (2011)) where number of states were limited based on a new definition of states space which solved the mobile robot navigation problem in dynamic environment.

Apart from Q-learning RL algorithm for path planning of mobile robots, one more RL algorithm, called deep deterministic policy gradient (DDPG) is also explored for this purpose. DDPG algorithm for single robot is extended to Parallel DDPG algorithm for multi-robot system for construction and navigation task (Chen et al. (2019a)). For autonomous unmanned aerial vehicles (UAVs) landing on a moving platform, an algorithm based on deep reinforcement learning (DRL) was provided by (Rodriguez-Ramos et al. (2018)). As per this research, for continuous action and state domains, the DDPG algorithm provides prominent results. For mobile robots, a method for real time path planning and obstacle avoidance is developed by (Syed et al. (2014)), which is based on an algorithm called guided autowave pulse coupled neural network (GAPCNN). This proposed work was an improved edition of the recently presented model called modified pulse coupled neural network (MPCNN) by considering directional autowave control and dynamic thresholding technique to accelerate the firing of neurons. To control the motion of a robotic salamander, an RL algorithm produced a central pattern generator (CPG) model that generates rhythmic motion (Cho et al. (2019)). Since action spaces here are continuous, therefore they have proposed DDPG algorithm along with actor-critic method. Additional works include a four-wheeled omni-directional mobile robot, which was analysed for mapping and navigation based on robot operating system (ROS) (Quang et al. (2019)).

Since robots generally provide services in environments like warehouses, restaurants, hospitals, and houses, we have limited free space for their movements. So, we need a robot that can work in such restricted places easily and efficiently. This motivates us to work on omni-wheel robots, which is a holonomic drive and has better moving flexibility than nonholonomic drives. Its capability to rotate its own vertical axis and move in any oblique direction makes it more reliable than other wheel robots. Also, nowadays, an idea has been started in which omni-wheels are incorporated in vehicles which would be very helpful in the parking of vehicles in limited spaces. Therefore, the current work focuses on a path planning algorithm for omni-directional robotic platform.
The other popular RL algorithm used for a static environment is SARSA (State-Action-Reward State-Action). Compared to SARSA, the Q-learning algorithm takes less computation time and fewer steps to reach the desired target location (Sutton and Barto (2018); Sichkar (2019)). Since the Q-learning algorithm directly learns the optimal policy, it is also a good choice if we train our robot in a simulation environment. In earlier published studies the Q-learning was implemented for a very small grid world static environment (Nair and Supriya (2018); Harwin and Supriya (2019)) and most of the studies were focused on differential drive kind of robotic platform for path planning. In our proposed research, we have considered a quite large home environment with an obstacle density of around 45%, and here we have considered a holonomic mobile platform that has three omni-wheels for autonomous navigation.
The proposed research work has the following contributions: -

1. Q-learning algorithm provides a safe and shortest path to the mobile platform to navigate through multiple static obstacles of distinct shapes and at different positions in a home environment.
2. We improved the performance by training the Q-learning agent by eliminating the situation of getting into a small loop and assigning high penalty values to avoid double steps and to go outside the environment boundary.
3. By using the curve fitting technique, we have obtained the optimized trajectories.
4. The kinematic control of the omni-wheel robot is done using the CLIK algorithm.

The rest of this paper is organized as follows. Section 2 presents the methodology. Section 3 deals with the problem formulation for this proposed work. Finally, in Sections 4 and 5, detailed simulation results and conclusions respectively are discussed.

2. METHODOLOGY

This section describes the path planning and control algorithm used for the suggested work in detail.

2.1. Q-learning Algorithm

Several RL algorithms have been implemented on various categories of differential drive robots in the last few years. However, such algorithms implementation on three omni-wheel robots is a less studied field. In RL, the agent (learning system) acts in the environment and learns by trial and error to maximize its pleasure (+ reward) and minimize its pain (- reward). A policy defines what action the agent should choose when it is in a given situation. If the state and action spaces for the environment are discrete and few in number, we can represent a policy with a table called Q-table. Figure 1 shows the steps used in Q-learning algorithm. The Q-function Q(s, a) is updated using well known Bellman equation (Low et al. (2019)).

$$\text{New } Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma maxQ'(s',a') - Q(s,a)]$$

Or

$$\text{New } Q(s,a) = (1-\alpha)Q(s,a)+\alpha[R(s,a)+\gamma maxQ'(s',a')] \quad (1)$$

where, Q(s, a): current Q-value, $maxQ'(s', a')$: maximum expected future reward given the new state (s') and all possible actions at that new state, α: learning rate, γ: discounting factor for future rewards and R(s, a): reward for taking an action 'a' in a state 's'.

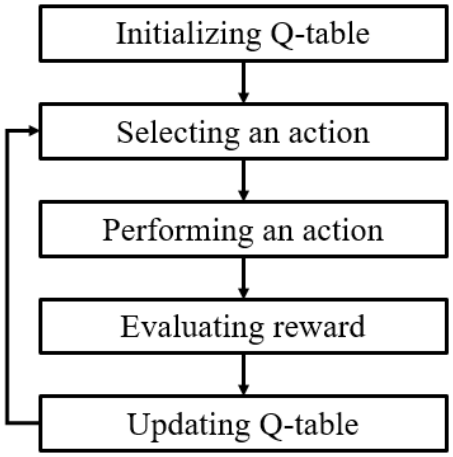

Figure 1. Steps in Q-learning.

2.2. Closed-loop Inverse Kinematics Algorithm

The closed-loop inverse kinematics (CLIK) algorithm is a numerical method used to approximate the solution of the inverse kinematics problem of robot manipulators based on the explicit Euler integration that is a simple numerical integration technique.

Conventional Inverse Kinematics (IK) using inverse Jacobian is given by (Rayankula and Pathak (2021)): -

$$\dot{q} = J^{-1}(q)\dot{X}_{CM} \quad (2)$$

Where, q, $\dot{q}$ and $\dot{X}_{CM}$ are joint angle, joint velocity, and velocity vector of center of mass (CM) of mobile base (MB) respectively. $J^{-1}(q)$ is the inverse matrix of $J(q)$. Here, Euler numerical integration implementation is given by,

$$q(t_{k+1}) = q(t_k) + J^{-1}(q)\dot{X}_{CM}(t) \triangle t \quad (3)$$

The error in the CM of mobile base position (E_{CM}) may be defined as,

$$E_{CM} = X_{ref} - X_{CM} \quad (4)$$

Where X_{ref} is reference trajectory of CM of mobile robot. The time derivative of E_{CM} is,

$$\dot{E}_{CM} = \dot{X}_{ref} - J(q)\dot{q} \quad (5)$$

To ensure convergence of error to zero, a relationship is established between $\dot{q}$ and E_{CM}, which is defined by CLIK algorithm,

$$\dot{q} = J^{-1}(q)[\dot{X}_{ref} + KE_{CM}] \quad (6)$$

Where, K is a positive definite gain matrix and choice of K guarantees that the error uniformly converges to zero.

3. PROBLEM FORMULATION

This section explains the indoor environment used for the navigation and kinematics modelling of the omni-wheeled platform.

3.1. Home Environment

It is a 20×20 cells indoor home environment, which includes a kitchen, a bathroom, a living room, and two bedrooms (A and B here). The map of the environment is generated using colormap, which displays the image with scaled colors as shown in Figure 2. Here black color represents the obstacles (-100 value in colorbar), and white space represents the free space (1 value in colorbar).

3.2. Omni-wheeled Robot

The mobile base consists of three omni-wheels which are 120 degrees apart. The omni-wheels have small rollers around the circumference. The effect of these rollers is that the wheels can slide easily in a lateral direction also. Such kinds of wheels are usually used in holonomic drives.

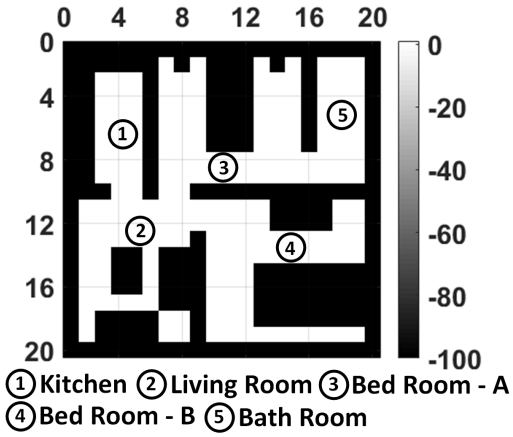

Figure 2. Indoor environment map (20×20 cells).

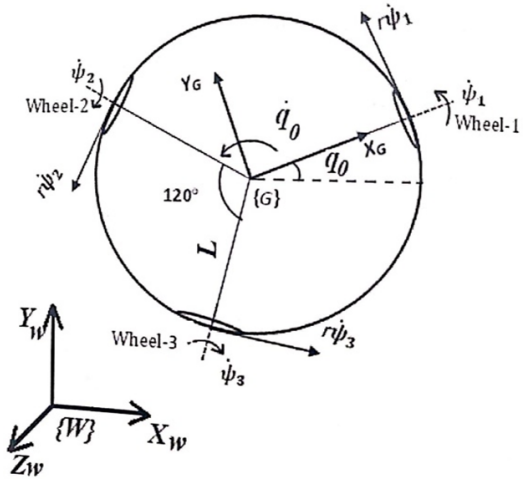

Figure 3. Velocity diagram of MB (Ram et al. (2019)).

Kinematics modelling: In Figure 3, let $\dot{q}_{x,w}$ and $\dot{q}_{y,w}$ represent the linear velocities of CM of MB in the X_W and Y_W directions in world reference frame $\{W\}$. $\dot{q}_{x,g}$ and $\dot{q}_{y,g}$ are the linear velocities of CM along X_G and Y_G directions in body reference frame $\{G\}$. $\dot{q}_o$ is rotational velocity of MB about a vertical axis passing through its CM. Angular velocities of the wheels 1, 2, and 3 about wheel axis are represented by $\dot{\psi}_1$, $\dot{\psi}_2$ and $\dot{\psi}_3$ respectively. Let r be the radius of each wheel and L be the radius of MB (Ram et al. (2019)).

The kinematic relations among CM velocities and wheel velocities of MB in matrix form can be written as,

$$\begin{bmatrix}\dot{\psi}_1\\ \dot{\psi}_2\\ \dot{\psi}_3\end{bmatrix} = 1/r\begin{bmatrix}0 & 1 & L\\ -\sqrt{3}/2 & -1/2 & L\\ \sqrt{3}/2 & -1/2 & L\end{bmatrix}\begin{bmatrix}\dot{q}_{x,g}\\ \dot{q}_{y,g}\\ \dot{q}_o\end{bmatrix} \tag{7}$$

The velocities of CM of MB in frame $\{G\}$ and world frame $\{W\}$ are related by,

$$\begin{bmatrix}\dot{q}_{x,w}\\ \dot{q}_{y,w}\\ \dot{q}_o\end{bmatrix} = \begin{bmatrix}\cos q_o & -\sin q_o & 0\\ \sin q_o & \cos q_o & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{q}_{x,g}\\ \dot{q}_{y,g}\\ \dot{q}_o\end{bmatrix} \tag{8}$$

On combining above two equations,

$$\begin{bmatrix}\dot{\psi}_1\\ \dot{\psi}_2\\ \dot{\psi}_3\end{bmatrix} = 1/r\begin{bmatrix}0 & 1 & L\\ -\sqrt{3}/2 & -1/2 & L\\ \sqrt{3}/2 & -1/2 & L\end{bmatrix}\begin{bmatrix}\cos q_o & \sin q_o & 0\\ -\sin q_o & \cos q_o & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{q}_{x,w}\\ \dot{q}_{y,w}\\ \dot{q}_o\end{bmatrix} \tag{9}$$

Finally we get,

$$\begin{bmatrix}\dot{\psi}_1\\ \dot{\psi}_2\\ \dot{\psi}_3\end{bmatrix} = 1/r\begin{bmatrix}-\sin q_o & \cos q_o & L\\ -\sin(pi/3 - q_o) & -\cos(pi/3 - q_o) & L\\ \sin(pi/3 + q_o) & -\cos(pi/3 + q_o) & L\end{bmatrix}\begin{bmatrix}\dot{q}_{x,w}\\ \dot{q}_{y,w}\\ \dot{q}_o\end{bmatrix} \tag{10}$$

The above equation can be rewritten as,

Table 1. Parameters for Q-learning Algorithm

Parameter	Value
Reward for free space	1
Reward for obstacles	-100
Reward for start point (SP)	1
Reward for goal point (GP)	+25
Reward for outside the environment boundary	$-\infty$
Number of Iterations	100
Max number of total steps	400
Discount factor (γ)	0.90
Learning rate (α)	0.50

$$\begin{bmatrix}\dot{q}_{x,w}\\ \dot{q}_{y,w}\\ \dot{q}_o\end{bmatrix} = [J_{MB}]\begin{bmatrix}\dot{\psi}_1\\ \dot{\psi}_2\\ \dot{\psi}_3\end{bmatrix} \tag{11}$$

where, $[J_{MB}]$ be the mobile base Jacobian and is defined as,

$$[J_{MB}] = r/3\begin{bmatrix}-2\sin q_o & -2\sin(pi/3 - q_o) & 2\sin(pi/3 + q_o)\\ 2\cos q_o & -2\cos(pi/3 - q_o) & -2\cos(pi/3 + q_o)\\ 1/L & 1/L & 1/L\end{bmatrix} \tag{12}$$

4. RESULTS

4.1. Creation of Safe and Shortest Path using Q-Learning

To create the safe and shortest path using Q-learning the following parameters are considered: -
States: Here each cell (of size 1 feet $\times$ 1 feet), represents a state. So, there are total 400 states.
Actions: There are 8 possible actions at any given state: left, right, up, down and 4 oblique movements. For straight movements the path length is 1 feet and for oblique movements it is 1.4 feet. Other parameters used in Q-learning are enumerated in Table 1.

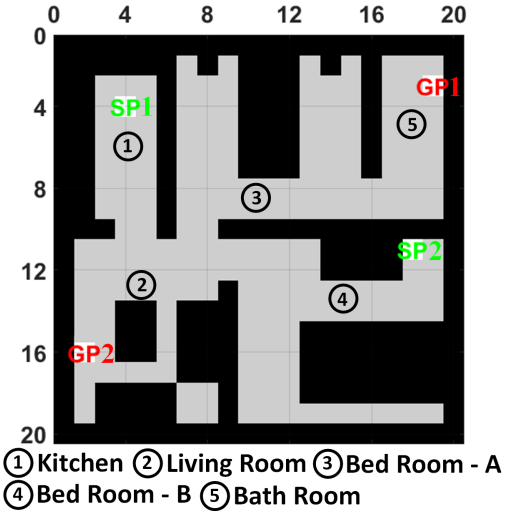

Figure 4. Location of SP and GP for both the cases.

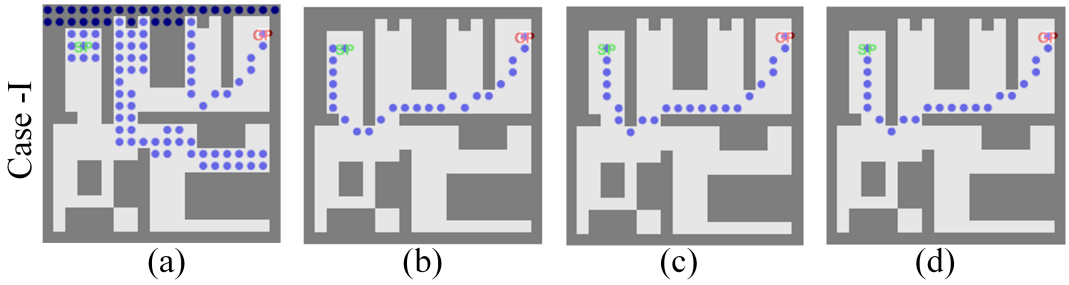

Figure 5. Q-learning training for Case-I (a) After 10 iterations (b) After 20 iterations (c) After 30 iterations (d) After 40 iterations.

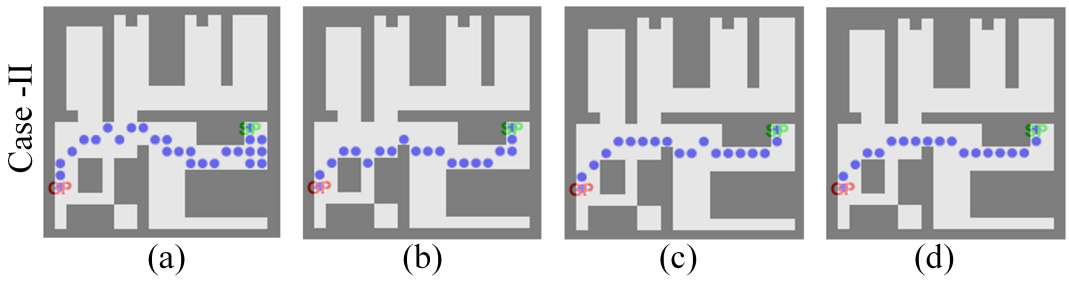

Figure 6. Q-learning training for Case-II (a) After 10 iterations (b) After 20 iterations (c) After 30 iterations (d) After 40 iterations.

The current work focuses on the simulation of omni-directional mobile robot for two different cases. The home environment for the robot with SP and GP is shown in Figure 4 with 1 and 2 suffix for Case-I and Case-II respectively. SP1 is in kitchen (4,4 Cell) and GP1 is in bathroom (19,3 Cell). While, SP2 is in bedroom-B (18,11 Cell) and GP2 is in living room (2,16 Cell).

For an unknown environment, the robot first learns about the obstacles and the free space with the help of Q-learning training algorithm. After several iterations, the robot finds the safe and shortest path from SP to GP. The training for first 40 iterations with an interval of 10 iterations for Case-I and Case-II is shown in Figure 5 and Figure 6 respectively.

After successful training with 100 iterations, the robot is able to find the safe and shortest path between two set points. The optimal path for both the cases is shown in Figure 7.

The path length covered by the robot to reach the GP can observed in Figure 8. In the initial iterations the path covered by the robot is longer and as the robot learns more and more about the environment the path length decreases with more iterations. After 100 iterations the path length and steps taken for Case-I and Case-II are 25.2 feet, 23 steps and 20 feet, 19 steps respectively.

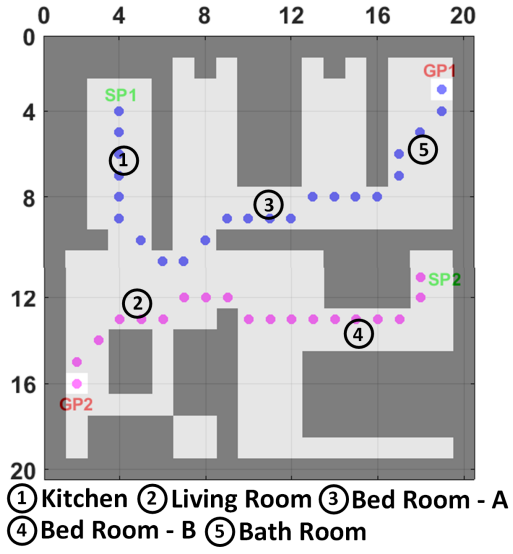

Figure 7. Final optimal path for both the cases after successful training.

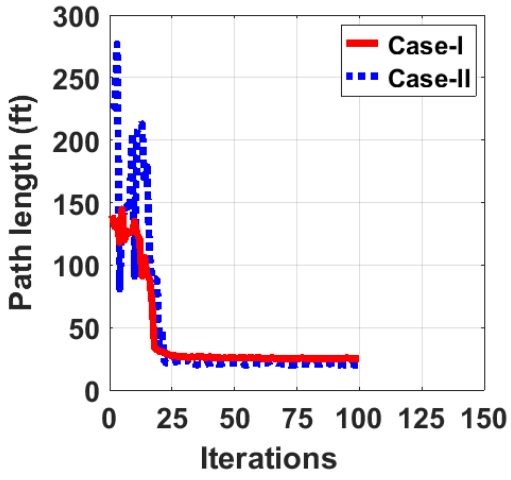

Figure 8. Path length vs iterations graph for both the cases.

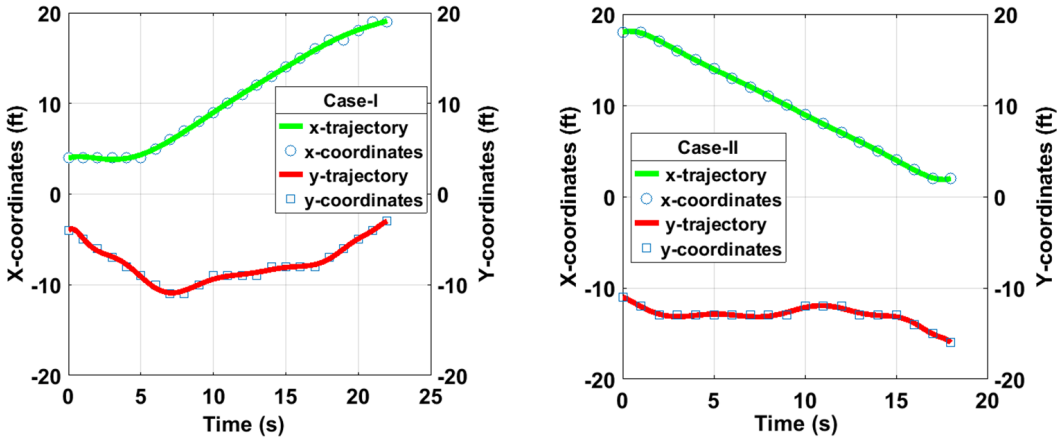

Figure 9. Trajectory vs time plot for both the cases.

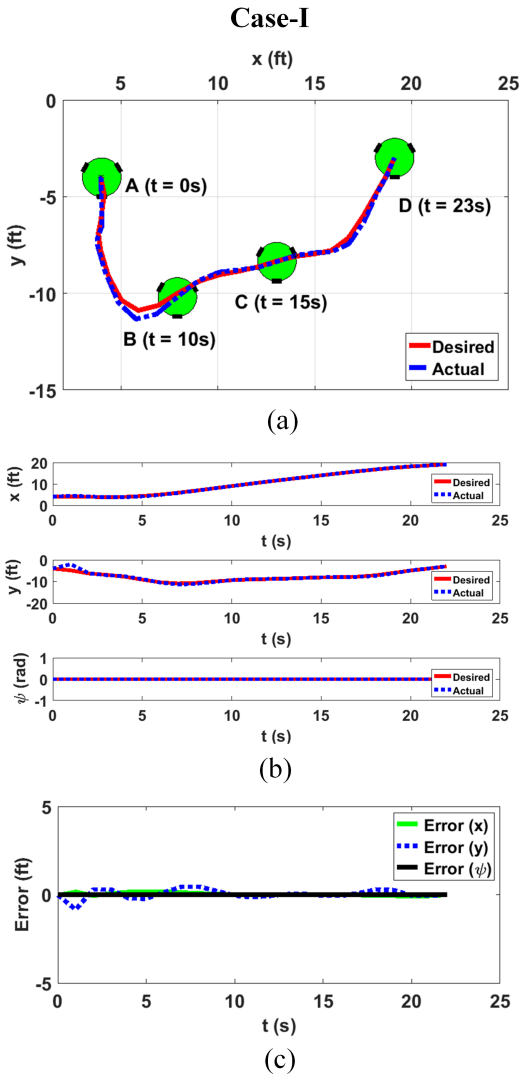

Figure 10. Plots of desired vs actual trajectories of omni-wheeled robot for Case-I (a) x vs y (b) Trajectory vs time (c) Trajectory error vs time.

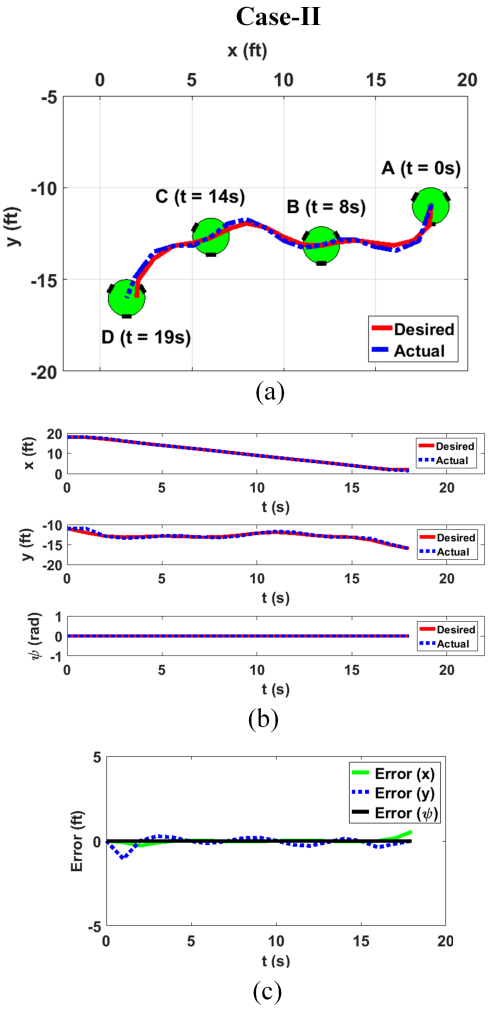

Figure 11. Plots of desired vs actual trajectories of omni-wheeled robot for Case-II (a) x vs y (b) Trajectory vs time (c) Trajectory error vs time.

4.2. Trajectory Generation using Curve Fitting

After obtaining the safe and shortest path by Q-learning, we extract the x and y coordinates from that path and then best suited polynomial curve fitting is used to generate the trajectories (x-t and y-t trajectories) as shown in Figure 9.

4.3. Path Planning of Omni-wheeled Robot using CLIK

After getting the trajectories for both cases from polynomial curve fitting, the CLIK algorithm is used for kinematic control of the omni-wheel robot to follow the desired safe and shortest path. The comparison between desired and actual trajectory followed by the robot can be seen in Figure 10 and Figure 11 for the considered two cases. Figure 10 (a) and Figure 11 (a) shows the desired path and the actual path followed by the robot. In these figures, to track the path of the robot four positions of the robot (A, B, C, and D) are shown at different time intervals. Figure 10 (b) and Figure 11 (b) shows the desired and actual trajectory of the robot such that position (x and y) and orientation (ψ) of CM of MB with time and finally the error in position and error in orientation of CM of MB with time is shown in Figure 10 (c) and Figure 11 (c). It can be clearly observed that the robot is able to follow the desired path with significant accuracy and it deviates slightly from its desired path when there is a curvature in the path. The use of superior control model can avert this inadequacy.

5. CONCLUSION

The current research focuses on implementing an RL algorithm called Q-learning on three wheels omni-directional robot. The work demonstrates Q-learning applicability with two different cases for a home environment with different start and goal points. The safe and shortest path is explored using the algorithm by training the robot in a world with static obstacles of different shapes and sizes. Afterward, the CLIK algorithm is used for the kinematic control of the omni-wheeled mobile robot to follow the explored optimal path. The main conclusions from the current work are (1) the Q-learning algorithm is able to produce applicable results to generate an optimal path to be followed by the omni-wheeled robot. (2) The simulation results show that the robot is able to follow the explored optimal path with significant accuracy in both cases. (3) The simulation practitioners can get an idea of how to create a custom environment for simulation in MATLAB. They can visualize the Q-learning training process and optimal path graphically.

The Q-learning algorithm is best suited for a static environment and when the state and action spaces for the environment are discrete and few in number. Therefore, in general, the limitation of Q-learning is that it is not able to recognize the dynamic obstacles and does not perform well when the state space is large and the action space is continuous. In future works, we hope to implement deep reinforcement learning (DRL) algorithms that are efficient in tackling complex environments with both static and dynamic obstacles. The work will also be extended for experimentation to test the applicability of algorithms in real-time environments.

References

Chen, W., Zhou, S., Pan, Z., Zheng, H., and Liu, Y. (2019a). Mapless collaborative navigation for a multi-robot system based on the deep reinforcement learning. *Applied Sciences*, 9(20):4198.

Chen, Y., Dong, C., Palanisamy, P., Mudalige, P., Muelling, K., and Dolan, J. M. (2019b). Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

Cho, Y., Manzoor, S., and Choi, Y. (2019). Adaptation to environmental change using reinforcement learning for robotic salamander. *Intelligent Service Robotics*, 12:209–218.

Duguleana, M. and Mogan, G. (2016). Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems with Applications*, 62:104–115.

Harwin, L. and Supriya, P. (2019). Comparison of sarsa algorithm and temporal difference learning algorithm for robotic path planning for static obstacles. In *2019 Third International Conference on Inventive Systems and Control (ICISC)*, pages 472–476. IEEE.

Jaradat, M. A. K., Al-Rousan, M., and Quadan, L. (2011). Reinforcement based mobile robot navigation in dynamic environment. *Robotics and Computer-Integrated Manufacturing*, 27:135–149.

Low, E. S., Ong, P., and Cheah, K. C. (2019). Solving the optimal path planning of a mobile robot using improved q-learning. *Robotics and Autonomous Systems*, 115:143–161.

Manh, T. N., Manh, C. N., Tien, D. P., Van, M. T., Kim, D. H. T., and Duc, D. N. (2020). Autonomous navigation for omnidirectional robot based on deep reinforcement learning. *International Journal of Mechanical Engineering and Robotics Research*, 9(8):1134–1139.

Nair, D. S. and Supriya, P. (2018). Comparison of temporal difference learning algorithm and dijkstra's algorithm for robotic path planning. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1619–1624. IEEE.

Quang, H. D., Manh, T. N., Manh, C. N., Tien, D. P., Van, M. T., Kim, D. H. T., Van, N. T. T., and Duan, D. H. (2019). Mapping and navigation with four-wheeled omnidirectional mobile robot based on robot operating system. *Proceedings of the 2019 International Conference on Mechatronics, Robotics and Systems Engineering, MoRSE 2019*, pages 54–59.

Ram, R., Pathak, P. M., and Junco, S. J. (2019). Trajectory control of a mobile manipulator in the presence of base

disturbance. *Simulation*, 95(6):529–543.

Rayankula, V. and Pathak, P. M. (2021). Fault tolerant control and reconfiguration of mobile manipulator. *Journal of Intelligent & Robotic Systems*, 101(2):1–18.

Rodriguez-Ramos, A., Sampedro, C., Bavle, H., Moreno, I. G., and Campoy, P. (2018). A deep reinforcement learning strategy for uav autonomous landing on a moving platform alejandro. *IEEE International Conference on Intelligent Robots and Systems*, pages 1010–1017.

Sichkar, V. N. (2019). Reinforcement learning algorithms in global path planning for mobile robot. In *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 1–5. IEEE.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

Syed, U. A., Kunwar, F., and Iqbal, M. (2014). Guided autowave pulse coupled neural network (gapcnn) based real time path planning and an obstacle avoidance scheme for mobile robots. *Robotics and Autonomous Systems*, 62:474–486.

Wang, P., Li, X., Song, C., and Zhai, S. (2020). Research on dynamic path planning of wheeled robot based on deep reinforcement learning on the slope ground. *Journal of Robotics*, 2020.

Wen, S., Zhao, Y., Yuan, X., Wang, Z., Zhang, D., and Manfredi, L. (2020). Path planning for active slam based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 13(2):263–272.

Yin, S., Ji, W., and Wang, L. (2019). A machine learning based energy efficient trajectory planning approach for industrial robots. *Procedia CIRP*, 81:429–434.

Yu, J., Su, Y., and Liao, Y. (2020). The path planning of mobile robot by neural networks and hierarchical reinforcement learning. *Frontiers in Neurorobotics*, page 63.