Implementation and Analysis of Modified SARSA algorithm for Robotic Path Planning:

Laya Harwin[1] and **Supriya P**[2]

[1]Department of Electrical and Electronics Engineering
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India:
Email: layaamy.harwin@gmail.com

[2]Department of Electrical and Electronics Engineering
Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India:
Email: p_supriya@cb.amrita.edu

ABSTRACT

The involvement of robots in our day-to-day life is making our lives much easier. They should be employed with the potential to move autonomously in an unknown environment. They should also be capable of navigating all the static and dynamic obstacles and reaching the final point safely. In order to do that, pathfinding is a fundamental requirement. In this paper, the SARSA algorithm is studied, modified, and implemented for an 8x8 grid to determine the optimal path from the source point to the destination point for both static and dynamic obstacles in MATLAB and in mbed microcontroller interfaced with iRobot create with an android application for interaction with the user and a GUI for displaying the path traveled.

Keywords: Modified SARSA Algorithm, Path Planning, iRobot Create.

Mathematics Subject Classification: 00-02

1. INTRODUCTION

Robotics is a versatile and emerging area influencing all our lives. The ability of robots to work without rest will replace human beings at work. An autonomous robot should have the capability to understand the characteristics of its surroundings and should be able to inherit necessary information from the environment which enables them to be user-friendly. After gaining the essential data from the surrounding, it needs to trace its path to its destination. This is called path planning.

Path planning is a necessity for autonomous robots which helps these robots to track their path from the source grid to the destination grid. Numerous path-planning algorithms have been proposed so far. The selection of a path-planning algorithm depends on its application. One of the reinforcement learning (RL) algorithms known as the SARSA algorithm is studied and modified in this paper. The full form of SARSA is state, action, reward, next state, next action.

The paper is organized into five sections. The first portion introduces the importance of robotics and path planning. The second section deals with the related work on path planning, Reinforcement Learning, Android application, the occurrence of dynamic obstacles, and the SARSA algorithm. The problem formulation of the modified SARSA algorithm comes in the third section. Simulation results are highlighted in the fourth section. The fifth section is for the implementation results and the final section contains the conclusions and future scope of the related work.

2. RELATED WORK

A robot that is fully automated should be able to extract necessary information from the environment in different ways depending on the implemented application without the intercession of humans. Designing such robots that are self-governing in amorphous, dynamic, and onerous environments still remains a challenge [2]. All autonomous robots reach the endpoint through simultaneous localization, mapping, and path planning taking the most advantageous path [3] [10]. The provision of the topological map can be utilized for pathfinding [8]. A variety of path-planning algorithms have been designed with time which helps these robots to find the ideal path [1].

According to the type of environment in which the robot is installed, path planning will be done either for static or dynamic obstacles and for the known or unknown environment [7]. Path planning algorithm that desires lesser time with multiple robots can aim ant colony optimization algorithm [4]. Machine learning algorithms are also used for path planning like temporal difference algorithm which designates rewards according to the direction of movement [6]. Path planning algorithms with fuzzy methods are put in action where the accuracy can be increased with more inputs [5].

Real-time employment applications with unceasing updates from the surroundings can use reinforcement learning algorithm, which is a type of machine learning algorithm and also a branch of artificial intelligence [7][15]. Reinforcement learning can be of two types. They are model-based RL and model-free RL. Awareness about the environment will be provided in model-based RL whereas no information will be given in model-free RL [1]. The primary concept of the RL algorithm is communication with the environment through learning [9]. Here the robots regularly balance their actions in order to obtain the ideal movement by increasing its performance [26]. It is found through the trial and error method [16][25].

There are different types of RL path planning algorithms from which the SARSA algorithm is studied and modified here. This learning algorithm is an on-policy algorithm. It has a wide range of applications and has proved its performance in various fields. It is used in the swarm RL algorithm in which numerous robots communicate with each other in order to interchange information [11]. In urban areas, it is used to control traffic which is self-adaptive [12]. It is also used in power systems to determine the optimal power in these systems [14]. It is used for channel allocation in cellular networks which permits call blocking [13]. The SARSA algorithm is also an approach to learning the markov decision process (MDP) [15][17].

Collision-free path planning has always been a constraint when both static and dynamic obstacles are present in the environment. In an environment where both unknown and moving obstacles are present fuzzy based path planning is used [18]. D* lite path planner is also used for path planning with dynamic obstacles in a haphazardly partitioned environment [19]. The occurrences of static obstacles are found with the help of ultrasonic sensors interfaced with a microcontroller in the AGO algorithm [4] and fuzzy algorithm [5].

The design of robots that are capable of moving from altering source and destination points is emerging with time. In [20], a prototype version of a customized shopping system was built where the information about the destination point is communicated through mobile applications by the customers, and the products will be distributed by the distribution centers. Android applications are used to get voice and gesture commands for smart robot assistant applications [21]. Android apps are easily created using applications like app inventor [22] and app inventor2 [23] which is an open-source tool [24]. These Android apps make it easy to communicate the position of obstacles to the robots.

3. PROBLEM FORMULATION

For increasing the performance of machines and software agents, the RL algorithm is used to find its own ideal behavior autonomously. The agents learn about their behavior through rewards provided to them which are known as reinforcement signals. The agents should take the best action from the current state to reach the next state. RL algorithm helps an agent to determine the suitable action according to the reward and punishment concepts. There are different algorithms that are available under reinforcement learning and the one focused here is the SARSA algorithm for robotic path planning. The algorithm is modified and made even simpler to implement and with lesser commutation.

3.1. Modified SARSA Algorithm

Modified SARSA is expanded as state-action-reward-next state-next action. It represents how the final Q function will be modified. It depends on the agent's current state "$\mathbf{S}_1$", the action chosen by the agent "$\mathbf{A}_1$", the reward given for choosing this action "**R**", the next state the agent reaches after the action is taken "$\mathbf{S}_2$", and finally the next action the agent chooses in the new state "$\mathbf{A}_2$". The acronym for the quintuple (s_t, a_t, r_t, s_{t+1}, a_{t+1}) is SARSA.

The steps involved in the modified SARSA algorithm are described below :

1) First, arbitrarily initialize Q [s, a]. Then the current state- s is observed and then an action is selected on Q randomly.
2) Then action a is done and then the reward and state- s' is observed.
3) Next action- a' is selected on Q by a policy.
4) Update the Q-value according to the equation:

$$Q[s,a] \leftarrow Q[s,a] + \alpha(r + \gamma Q[s',a']) \tag{3.1}$$

   where Q [s, a] is the current state of Q–value, Q [s', a'] is the next state of Q-value, r is the function for reward decided based on the policy, α, and γ are constants.
5) Then take the next state and next action.
   Repeat steps 2-5 until the destination grid is reached.

3.2. System Layout of 8x8 grid

The system layout of an 8×8 grid is shown in figure 1. The x and y coordinates of each grid are shown as (x, y). This layout can be extended for the N×N grid. This will help us to understand the position of

obstacles, the position of the source point, and the position of the destination point in the grid. The starting point will be (0.5,0.5) and the destination point will be (5.5,5.5). Source point and destination can be altered but, in most cases, it's taken as (0.5,0.5) and (7.5,7.5) for a better understanding of how the algorithm works for different positions of obstacles.

(0.5, 7.5)	(1.5, 7.5)	(2.5, 7.5)	(3.5, 7.5)	(4.5, 7.5)	(5.5, 7.5)	(6.5, 7.5)	(7.5, 7.5)
(0.5, 6.5)	(1.5, 6.5)	(2.5, 6.5)	(3.5, 6.5)	(4.5, 6.5)	(5.5, 6.5)	(6.5, 6.5)	(7.5, 6.5)
(0.5, 5.5)	(1.5, 5.5)	(2.5, 5.5)	(3.5, 5.5)	(4.5, 5.5)	(5.5, 5.5)	(6.5, 5.5)	(7.5, 5.5)
(0.5, 4.5)	(1.5, 4.5)	(2.5, 4.5)	(3.5, 4.5)	(4.5, 4.5)	(5.5, 4.5)	(6.5, 4.5)	(7.5, 4.5)
(0.5, 3.5)	(1.5, 3.5)	(2.5, 3.5)	(3.5, 3.5)	(4.5, 3.5)	(5.5, 3.5)	(6.5, 3.5)	(7.5, 3.5)
(0.5, 2.5)	(1.5, 2.5)	(2.5, 2.5)	(3.5, 2.5)	(4.5, 2.5)	(5.5, 2.5)	(6.5, 2.5)	(7.5, 2.5)
(0.5, 1.5)	(1.5, 1.5)	(2.5, 1.5)	(3.5, 1.5)	(4.5, 1.5)	(5.5, 1.5)	(6.5, 1.5)	(7.5, 1.5)
(0.5, 0.5)	(1.5, 0.5)	(2.5, 0.5)	(3.5, 0.5)	(4.5, 0.5)	(5.5, 0.5)	(6.5, 0.5)	(7.5, 0.5)

Figure 1. System Layout

Grid setup for an 8x8 grid is created for hardware testing with squares of length 40cm. The grid size is chosen as 40 cm so that the iRobot platform on which the SARSA algorithm is implemented and tested operates with sufficient spacing. 64 such continuous grids will be created according to the system layout.

3.3. System Architecture

Figure 2 shows the system architecture of the hardware implementation part of the algorithm. iRobot is used to implement reinforcement learning-based path planning.

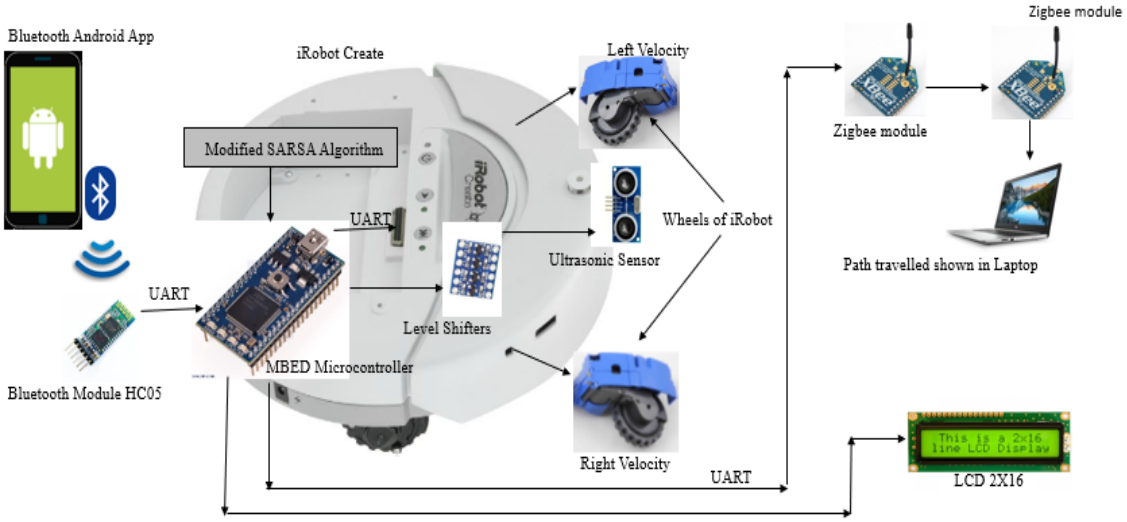

Figure 2. System Architecture of SARSA Algorithm

An Android-based Bluetooth app is created for communicating the number of static obstacles and their position of static obstacles. Bluetooth app communicates to MBED microcontroller through a Bluetooth module. The modified SARSA algorithm is implemented in the MBED microcontroller for path planning. Then according to the algorithm, the right and left wheels of the iRobot are controlled which enables the robot to move from one grid to the other. For detecting dynamic obstacles, an ultrasonic sensor is interfaced with the MBED microcontroller. After a dynamic obstacle is detected, the algorithm notes the position of the dynamic obstacle and reroutes its path to the destination from the current point. After the robot reaches its destination, an LCD is used to display a message stating that the robot has reached its destination. It also displays the time taken to travel and the distance traveled by the robot. MBED is also interfaced with a ZigBee module which communicates the path traveled to a computer that displays the path taken in a GUI.

4. SIMULATION RESULTS

Firstly, modified SARSA algorithm was implemented in MATLAB R2018b for an 8x8 grid for static obstacles.

4.1. Assumptions Made

Some of the assumptions made before implementing in MATLAB are the following :

1) An 8x8 grid containing 64 grids is used for path planning
2) Obstacles are assumed to be placed at the center of the grid.

4.2. Path Planning for different obstacle occurrence scenarios

Path planning was done for different numbers of obstacles from 1 to 30 and the results of a few cases are portrayed. The robot tends to take diagonal movement when there are no obstacles since it gives the optimal path rather than taking a vertical or horizontal movement. Some of the cases are highlighted here for different number of static obstacles. Figure 3(a) shows the path planning with one

static obstacle in the position (1.5,1.5) represented in a red square box. The source point is at (0.5,0.5) represented in a white square box and the destination point is at (7.5,7.5) represented in a blue square box. Figure 3(b) shows the path planning with five static obstacles with the same source and destination point as that of figure 3(a). Path planning with 10, 15, 20, and 25 static obstacles placed randomly is shown in figure 3(c),3(d),3(e), and 3(f) respectively.

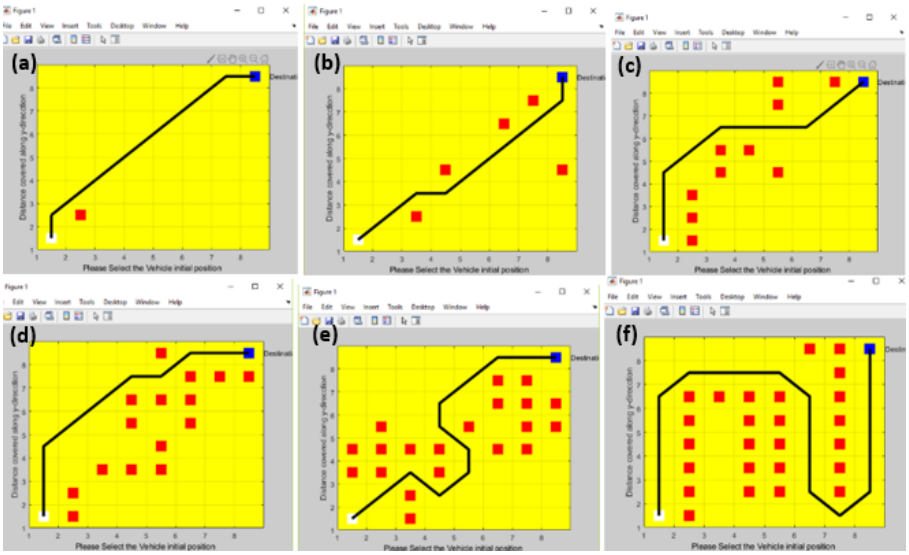

Figure 3. (a) Path planning with one static obstacle, (b) Path planning with five static obstacles, (c) Path planning with 10 static obstacles, (d) Path planning with 15 static obstacles, (e) Path planning with 20 static obstacles, (f) Path planning with 25 static obstacles

4.3. Path Planning with altering source and destination points

Path planning is done by altering source and destination positions other than (0.5,0.5) and (7.5,7.5) to verify the robustness of the algorithm. A few cases are highlighted here for different number of static obstacles. Figure 4(a) shows the path planning with 5 static obstacles represented in a red square box. The source point is at (0.5,3.5) represented in a white square box and the destination point is at (5.5,4.5) represented in a blue square box. Figure 4(b) shows the path planning with 9 static obstacles. The source point is at (1.5,6.5) and the destination point is at (6.5,0.5). Figure 4(c) shows the path planning with 16 static obstacles. The source point is at (6.5,5.5) and the destination point is at (1.5,1.5). Figure 4(d) shows the path planning with 25 static obstacles. The source point is at (0.5,2.5) and the destination point is at (0.5,4.5).

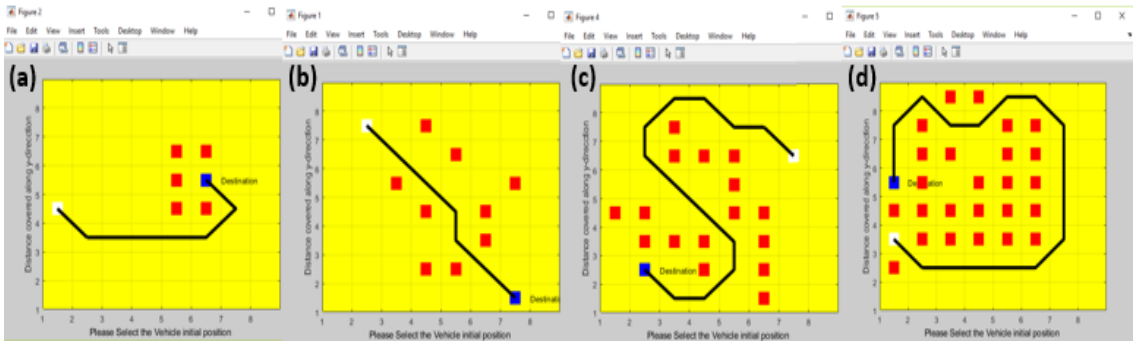

Figure 4. (a) Path planning with 5 static obstacles, (b) Path planning with 9 static obstacles, (c) Path planning with 16 static obstacles, (d) Path planning with 25 static obstacles

4.4. Cases where the algorithm fails

There are cases where the robot is unable to reach the destination point due to obstacles. These cases occur mainly when the source or destination points are covered up with obstacles and the robot is unable to proceed further. Figures 5(a) and 5(b) show the cases where the source and destination points have such obstacle positions and hence the algorithm fails. In such cases, a message stating no path exists is displayed as shown in figure 5(c).

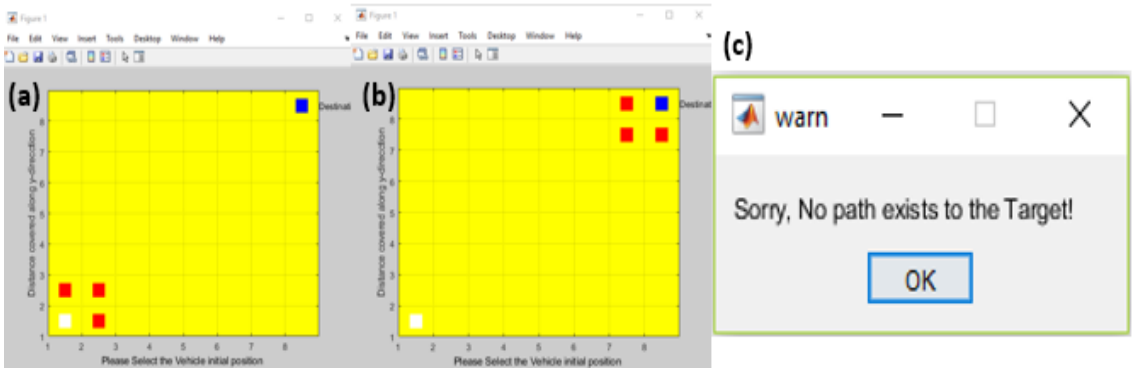

Figure 5. (a) Case 1 when the algorithm fails, (b)Case 2 when the algorithm fails, (c) Message displayed when the algorithm fails

4.5. Computation of time elapsed for different scenarios

The computation time of the algorithm only is evaluated in simulation. For different cases, the time is measured and recorded and is shown in table 1. The source point of all mentioned cases is (0.5,0.5) and the destination point is (7.5,7.5).

Table 1: Elapsed time for different obstacle occurring scenarios

SL NO	Number of Obstacles	Position of Obstacles	Elapsed time calculated in MATLAB in seconds
1	1	(4.5,4.5)	0.097362
2	3	(6.5,7.5), (6.5,6.5), (6.5,5.5)	0.150300
3	5	(1.5,0.5), (1.5,1.5), (1.5,2.5), (6.5,6.5), (6.5,7.5)	0.186294
4	7	(1.5,0.5), (1.5,1.5), (1.5,2.5), (2.5,3.5), (5.5,5.5), (6.5,6.5), (6.5,7.5)	0.217989
5	9	(1.5,0.5), (1.5,1.5), (1.5,2.5), (5.5,4.5), (5.5,3.5), (5.5,2.5), (5.5,6.5), (6.5,6.5), (6.5,7.5)	0.369123

5. IMPLEMENTATION RESULTS

For implementing modified SARSA in hardware, an ARM cortex M3 LPC1768 microcontroller is used which will perform the path planning according to the algorithm.

5.1. Cases tested in hardware with modified SARSA algorithm

Different cases have been tested in hardware with different number of obstacles with source point as (0.5,0.5) and destination point as (7.5,7.5). Some of the cases are shown in this section. In figure 6(a) path planning is done with one static obstacle at (4.5,3.5) represented as a red square. The blue arrows indicate the path taken from the source point to the destination point. SP is the source point, and DP is the destination point. The number of steps taken is numbered along with the arrows. The total number of steps taken from the source to the destination is numbered in the final grid i.e. the destination grid. In figure 6(b), path planning is done with three static obstacles at (5.5,6.5), (6.5,6.5), and (7.5,6.5). In figure 6(c), path planning is done with five static obstacles at (0.5,1.5), (1.5,1.5), (2.5,1.5), (3.5,1.5), and (4.5,1.5). In figure 6(d), path planning is done with seven static obstacles at (0.5,2.5), (1.5,2.5), (2.5,2.5), (3.5,2.5), (4.5,2.5), (5.5,5.5), and (6.5,5.5). In figure 6(e), path planning is done with nine static obstacles at (1.5,1.5), (1.5,2.5), (2.5,1.5), (3.5,3.5), (3.5,5.5), (5.5,5.5), (5.5,4.5), (6.5,3.5) and (6.5,6.5).

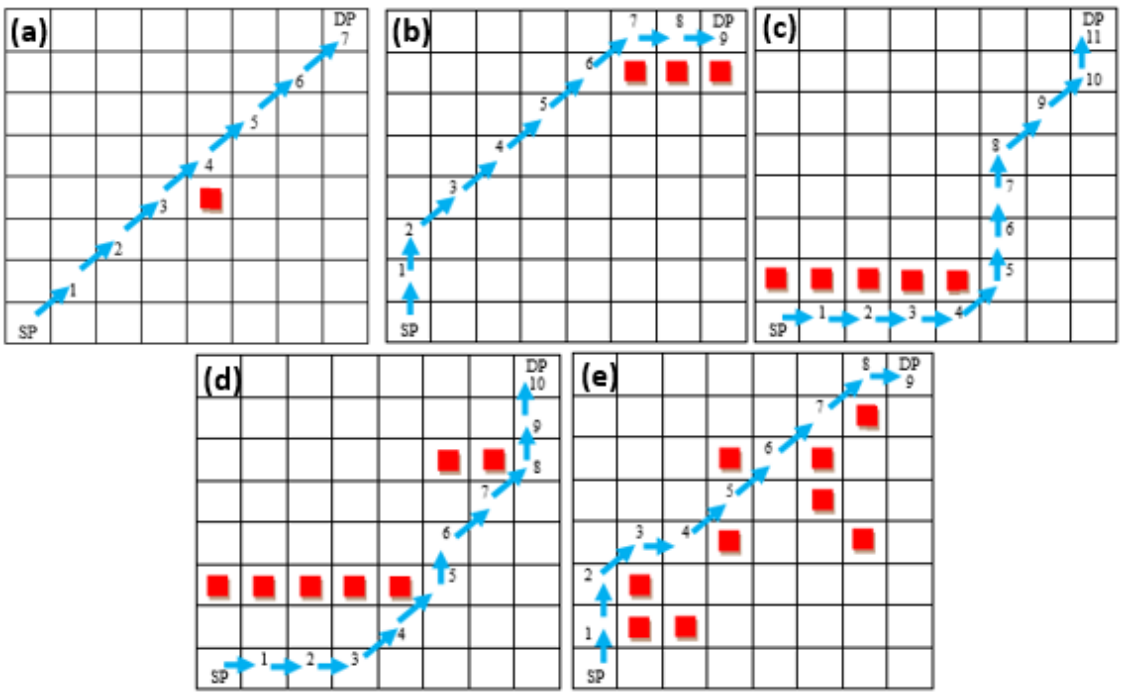

Figure 6. (a) Path planning with one static obstacle, (b) Path planning with 3 static obstacles, (c) Path planning with 5 static obstacles, (d) Path planning with 7 static obstacles, (e) Path planning with 9 static obstacles

5.2. Cases tested in hardware with modified SARSA algorithm for dynamic obstacles

Different cases were tested with more than one dynamic obstacle and a few of them are highlighted here. In figure 7(a), there are two static obstacles in the positions (1.5,1.5) and (2.5,2.5) and one dynamic obstacle in the position (4.5,5.5). Static obstacles are represented as red square boxes. Dynamic obstacles are represented as yellow square boxes. Blue arrows indicate the path taken before encountering a dynamic obstacle. When a dynamic obstacle is detected by an ultrasonic sensor, the algorithm runs again rerouting the path from the current point to the destination point marking the dynamic obstacles position. The black line indicates the path the robot takes after encountering a dynamic obstacle. In figure 7(b), there are two static obstacles in the position (0.5,6.5) and (0.5,7.5) and three dynamic obstacles in the positions (3.5,3.5), (4.5,5.5), and (6.5,6.5). Since

there are three dynamic obstacles, the algorithm reroutes the path three times from the current point where it encounters a dynamic obstacle. In figure 7(c), there are two static obstacles in the position (3.5,3.5) and (4.5,4.5) and five dynamic obstacles in the positions (0.5,1.5), (1.5,2.5), (2.5,3.5), (6.5,5.5) and (6.5,6.5). In this case, there are no blue arrows because the robot encounters a dynamic obstacle when it's in the initial position.

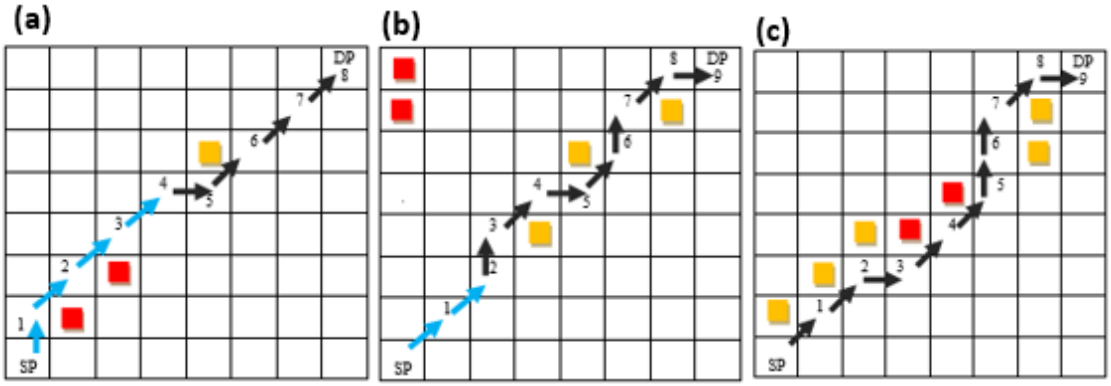

Figure 7. (a) Path planning with 2 static obstacles and 1 dynamic obstacle,
(b) Path planning with 2 static obstacles and 3 dynamic obstacles,
(c) Path planning with 2 static obstacles and 5 dynamic obstacles

5.3. Computation of the time elapsed and distance traveled for different scenarios in hardware

Different scenarios are tested in hardware and their time elapsed and distance traveled have been noted from LCD and is shown in Table II. The elapsed time is found using timers of the ARM controller. Elapsed time considers the total time including the time of computation of the algorithm, the time taken for the robot to reach the destination, and the time taken for communication. The distance is calculated after the robot takes each step and is cumulatively added till it reaches the destination. Figures 8(a) and 8(b) show the total time taken and total distance traveled displayed in LCD. The difference in time while comparing Table 1 and Table 2 is because Table I considers only the time of computation of the algorithm since it is a simulation of the algorithm whereas in Table 2 the time of computation of the algorithm and the time taken by the robot to reach the destination is considered.

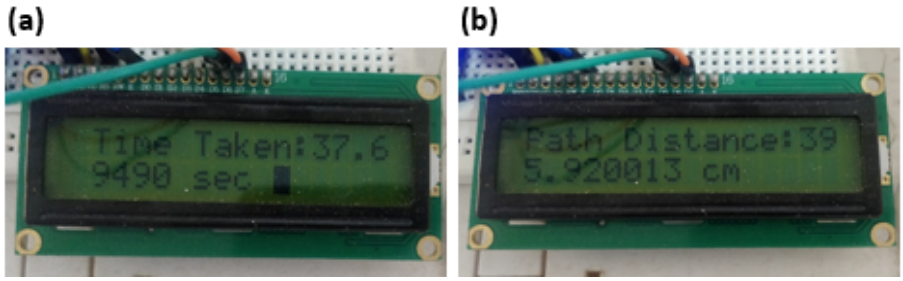

Figure 8. (a) Time taken displayed in LCD, (b)Total distance travelled displayed in LCD

Table 2: Elapsed time and distance travelled for different obstacle occurring scenarios

SL NO	Number of Obstacles	Position of Obstacles	Elapsed time calculated in hardware in seconds	Distance travelled calculated in hardware in cm
1	1	(4.5,4.5)	39.88	419.359
2	3	(6.5,7.5), (6.5,6.5), (6.5,5.5)	47.95	442.799
3	5	(1.5,0.5), (1.5,1.5), (1.5,2.5), (6.5,6.5), (6.5,7.5)	53.58	466.239
4	7	(1.5,0.5), (1.5,1.5), (1.5,2.5), (2.5,3.5), (5.5,5.5), (6.5,6.5), (6.5,7.5)	53.20	466.239
5	9	(1.5,0.5), (1.5,1.5), (1.5,2.5), (5.5,4.5), (5.5,3.5), (5.5,2.5), (5.5,6.5), (6.5,6.5), (6.5,7.5)	55.90	466.239

5.4. Path Planning for different obstacle occurrence scenarios

The 8x8 grid with 64 cells is shown in figure 9(a). The iRobot is at the source point at (0.5,0.5). The boxes are the obstacles. The robot moves from one grid to the other without hitting the boxes. In figure 9(b), iRobot create is shown which is interfaced with the ARM microcontroller. ARM microcontroller is interfaced with LCD, Bluetooth, ultrasonic sensor, and ZigBee module. The power supply for the ARM controller is given through power banks. The power for LCD and the ultrasonic sensor is taken from iRobot. The Bluetooth and ZigBee module is powered up using MBED. iRobot is powered up with a 14.4 V nickel metal hydride battery pack. Also, the app for 8x8 is shown in the figure with the iRobot. Figure 9(c) shows how the ultrasonic sensor is placed in iRobot so that it can correctly detect the dynamic obstacles that come its way. Figure 9(d) shows a close-up of the ARM controller, Bluetooth, and LCD connections. Figure 9(e), shows the ZigBee module which acts as the transmitter and sends data to the PC through ZigBee communication. Figure 9(f) shows the results obtained before the communication happens. The green color box indicates the source point, and the blue box indicates the destination point. In figure 9(g) the results obtained after the robot reaches the destination are shown which are received by the ZigBee module which acts as a receiver. This data is converted into a GUI using a processing tool. The red boxes indicate the path taken by the robot. The ZigBee module is also shown in the figure which is connected to the PC using a serial cable which is powered up by an external power supply.

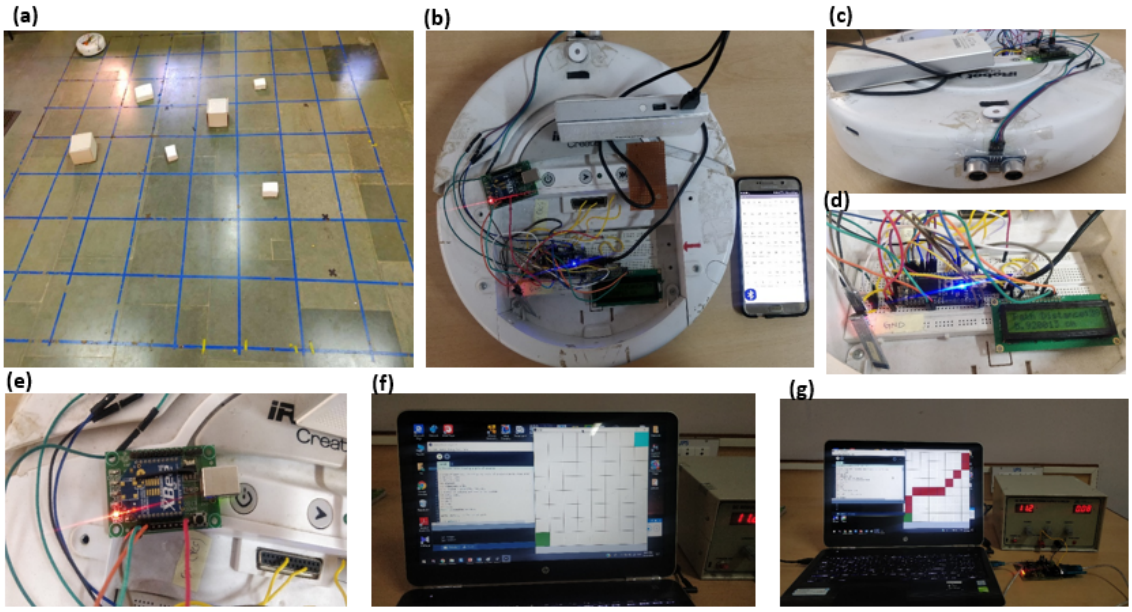

Figure 9. (a) Hardware Grid Setup with iRobot and obstacles, (b) iRobot and Android based app, (c) Placement of ultrasonic sensor in iRobot, (d) ARM controller interfaced with LCD and Bluetooth, (e) TX ZigBee, (f) GUI results before communication, (g) GUI results after communication

6. CONCLUSION AND FUTURE SCOPE

Simulation and implementation of the modified SARSA algorithm were done for different number of obstacles for an 8x8 grid. Different cases were tested where the source points and destination points are altering. Also, cases, where the algorithm fails, are discussed. Almost all the cases tested in MATLAB were successfully implemented in hardware. The cases where obstacles that are not defined as also detected and accordingly path planning was done. In all cases the robot reached the destination without colliding with the obstacles.

The work can be extended by using an IMU with an encoder which will localize the robot correctly and reduce the problems due to orientation and slipping. The modified SARSA algorithm can be used to solve complex mazes with more static and dynamic obstacles and with an increased size of the grid. It can also be implemented with different robots other than iRobot which can give more accurate results.

7. REFERENCES

1. Ravishankar N. R., Vijayakumar and M. V., 2017, Reinforcement Learning Algorithms : Survey and Classification, Indian Journal of Science and Technology, **10**, 0974-5645.

2. Xia, Chen, 2015, Intelligent Mobile Robot Learning in Autonomous Navigation, Automatic Control Engineering, Ecole Centrale de Lille.

3. Shreyas J, Sndeep J, 2017, Modern Machine Learning Approaches for Robotic Path Planning, International Journal of Computer Science and Information Technologies, **8**, 256-259

4. P. Joshy, P. Supriya, 2016, Implementation of Robotic path planning using Ant Colony Optimization Algorithm, International Conference on Inventive Computation Technologies, **1**, 1-6.

5. D. Davis, P. Supriya, 2016, Implementation of Fuzzy-Based Robotic Path Planning, 3rd IEEE International Conference on Engineering and Technology, **380**, 375-383.

6. Devika S. P. Supriya, 2018, Comparison of Temporal Difference Learning Algorithm and Dijkstra Algorithm for Robotic Path Planning, International Conference on Intelligent Computing and Control Systems,**10**, 1619-1624.

7. Zhang, Qian et al, 2012, Reinforcement Learning in Robot Path Optimization, Journal of Software, **7**, 657-662.

8. D. P. Romero-Martí, J. I. Núñez-Varela, C. Soubervielle-Montalvo and A. Orozco-de-la-Paz, 2016, Navigation and path planning using reinforcement learning for a Roomba robot, 2016 XVIII Congreso Mexicano de Robotica , 1-5.

9. Nihal Altuntas , Erkan Imal, Nahit Emanet, Ceyda Nur Ozturk, 2016, Reinforcement learning based mobile robot navigation, Turkish Journal of Electrical Engineering and Computer Sciences, **24**, 1747-1767

10. Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, Wolfram Burgard, 2017, Deep Reinforcement Learning with Successor Features for Navigation across Similar Environments, 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),2371-2378.

11. H. Iima, Y. Kuroe,2008, Swarm reinforcement learning algorithms based on Sarsa method, *2008 SICE Annual Conference*, Tokyo, 2045-2049

12. C. Li, M. Wang, S. Yang, Z. Zhang, 2009, Urban Traffic Signal Learning Control Using SARSA Algorithm Based on Adaptive RBF Network, 2009 International Conference on Measuring Technology and Mechatronics Automation, **3**, 658-661.

13. N. Lilith, K. Dogancay, 2005, Distributed reduced-state SARSA algorithm for dynamic channel allocation in cellular networks featuring traffic mobility, IEEE International Conference on Communications, **2**, 860-865.

14. M. R. Tousi, S. H. Hosseinian, A. H. Jadidinejad , M. B. Menhaj, 2008, Application of SARSA learning algorithm for reactive power control in power system, 2008 IEEE 2nd International Power and Energy Conference, 1198-1202.

15. Ganesha D, Venkatamuni, Vijayakumar Maragal, 2017, Implementation of modified SARSA learning technique in EMCAP, International Journal of Engineering & Technology, **7**,274-278.

16. Tran Xuan Sang, TranQuoc Kiet, Nguyen ThiUyen, 2017, Path Finding Algorithm for Autonomous Robots Based on Reinforcement Learning, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), **6**, 2278 – 1323

17. D. Xu, Y. Fang, Z. Zhang and Y. Meng, 2017, Path Planning Method Combining Depth Learning and Sarsa Algorithm, *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, **2**, 77-82.

18. Wei Wu, Zhang Qi Sen, J. B. Mbede, Huang Xinhan, 2001, Research on path planning for mobile robot among dynamic obstacles, *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, **2**, 763-767.

19. W. Yu, C. Yang, K. Su, Y. Tu, 2014, Dynamic path planning under randomly distributed obstacle environment, *2014 CACS International Automatic Control Conference (CACS 2014)*,138-143.

20. Mourtzis, Dimitris, Vlachou, Katerina, Zogopoulos, Vasilios, 2018, An IoT-based Platform for Automated Customized Shopping in Distributed Environments, 51st CIRP Conference on Manufacturing Systems, **72**, 892-897.

21. Libin M George, Annu Mariam Abraham, Ananthapadmanabhan J, Vineetha Anna Saji , Anil A R,2017, Implementation of IoT In Smart Robotics: A Survey, International Journal Of Engineering And Computer Science, **6**, 20762-20764.

22. Mackellar, Bonnie,2012, App inventor for android in a healthcare IT course, SIGITE'12 - Proceedings of the ACM Special Interest Group for Information Technology Education Conference. 245-250.

23. F. Turbak, D. Wolber, P. Medlock-Walton, 2014, The design of naming features in App Inventor 2, *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 129-132.

24. J. Tyler, 2011, *App Inventor for Android*, John Wiley & Sons Ltd, Th e Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

25. L. P. Kaelbling, M. L. Littman, and A. P. Moore, 1996, Reinforcementlearning: A survey,, Journal of Artificial Intelligence Research, **4**, 237–285.

26. Kwon, Woo Young, Suh, Il Hong, Lee, Sanghoon, Cho, Young-Jo,2007, Fast reinforcement learning using stochastic shortest paths for a mobile robot, 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, **1**, 82 - 87.