Reinforcement Learning based Automated Path Planning in Garden environment using Depth - RAPiG-D

Authors:
*S. Sathiya Murthi (Anna University, sathiyamurthi239@gmail.com)
Pranav Balakrishnan (Anna University, pranav.bk@outlook.com)
C Roshan Abraham (Anna University, charlroshan@gmail.com)
Dr. V. Sathiesh Kumar (Anna University, sathiieesh@gmail.com)

Abstract — Path planning by employing Reinforcement Learning is a versatile implementation that can account for the ability of a robot to autonomously map any unknown environment. In this paper, such a hardware implementation is proposed and tested by making use of the SARSA algorithm for path planning and by utilizing stereovision for depth estimation based obstacle detection. The robot is tested in a cell-based environment – 3x3 with 2 obstacles. The goal is to map the environment by detecting and mapping the obstacles and finding the ideal route to the destination. The robot starts at one end of the environment runs through it for a specified number of episodes and it is observed that the robot can accurately identify and map obstacles and find the shortest path to the destination in under 10 episodes. Currently the destination is a fixed point and is taken as the other diagonal end of the environment.

Keywords: Adaptive, Autonomous, Cell-based, Closed environment, Depth Estimation, Episodes, Depth map, Dynamic, Map, Micro-controller, Obstacle Detection, On Policy, Path planning, Q-Table, Reinforcement learning, Robot, Route, SARSA, Stereo Vision, Thresholding.

I. INTRODUCTION

Reinforcement Learning is an adaptable and learning based method that can allow an agent to learn and determine the optimal solution to a problem on its own based on the rewards and penalties offered by the environment [1]. This allows the agent to handle any state that it might encounter while learning to achieve the required results. SARSA or State-Action-Reward-State-Action is a basic Reinforcement Learning Algorithm, an On Policy technique. With increasing innovations and implementations of Reinforcement Learning in Self-driving vehicles [2], Autonomous Robots [3], UAVs [4][5] there rises a requirement of adaptive mapping of unknown environments and the ability to determine the shortest path to any given destination in order to handle unpredictable and dynamic environments. Reinforcement Learning is an effective method that can be used to satisfy the requirements of such an environment by learning from repeated iterations of trials to estimate the optimal path from source to destination while negotiating obstacles [6]. In this paper, this implementation has been employed to plan routes and identify obstacles in a garden environment which can be used to automate management and care taking of gardens and plants. This technique can further be extended to exploration and searching tasks such as Space exploration and mapping, Air crash and accident investigations.

II. RELATED WORKS

To discover the path between source and destination, Konor et al. reported on an enhanced Q-learning approach [7]. The step distance (from one state to the next) and the eventual destination are assumed here. It is used to update the entries in the Q-table. Unlike the traditional Q-learning approach, where the values are continually updated, the values are only entered once. At each state, the Q-value derived for the best action is saved. In terms of traversal time and the number of states traversed, performance tends to increase. [8] describes end-to-end path planning using Deep Reinforcement Learning. To estimate the Q-value for each state-action, a deep Q-network (DQN) is first created and trained. The RGB picture frame is fed into the DQN. The best course of action is chosen using an action selection approach. The authors claimed that using the DQN approach for path planning resulted in a successful outcome. Path planning is done out using a Q-learning algorithm based on the Markov Decision Process [9], according to Sichkar et al [10].

It is challenging to find an optimal path in complicated situations using the traditional Q-learning approach. The robot determined/identified an optimal path from the source to the destination by avoiding collisions with impediments in its propagation path, according to the authors. The shortest path between the source and destination is determined using Q-learning and SARSA algorithms [11]. The method has been tested in a simulated environment with preset barriers. Different learning periods are included in the two algorithms used. It also fluctuates in the number of steps it takes to get to its objective by avoiding collisions with objects along the route.

The shortest path between the source and destination cannot be found using traditional Breadth First Search (BFS) or Rapidly Exploring Random Trees (RRT) techniques. As a result, the authors designed and showed a path planning algorithm based on reinforcement learning [12]. To begin, a random route graph is chosen. If the chosen path has barriers, it is not taken into account. A collision-free route is found using the Q-learning approach. When compared to RRT and BFS algorithms, the suggested approach provided a smooth and quickest path, according to the authors.

In an unknown environment, the iterative SARSA algorithm [13] is used to discover the best path from the source to the destination. Traditional Reinforcement learning techniques are contrasted on criteria like route length and processing complexity (Q-learning and SARSA). The authors claim that as compared to typical Reinforcement learning approaches, the Iterative SARSA algorithm used during robot path planning produces better results.

Based on a thorough review of the literature, it has been determined that path planning Reinforcement learning is still in its infancy. The algorithm may be fine-tuned or improved further so that it can be used in real-time situations. The use of the Reinforcement learning algorithm in connection to path planning in an unknown environment is investigated in this work. In the process of picking a suitable action by the robot, the SARSA algorithm is applied.

III. METHODOLOGY

Reinforcement learning is a machine learning technique based on rewarding desired behaviors and/or penalizing undesired actions. A reinforcement learning agent is able to observe and interpret its environment, take actions and learn through experience.

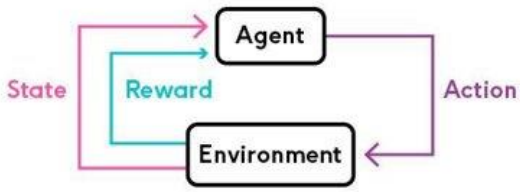

Figure 1: SARSA Illustration

Figure 1 depicts the basic flow of the SARSA algorithm: State – Action – Reward – State – Action. A software implementation of SARSA algorithm was initially tested to check validity of the path planning algorithm and its ability to map an unknown environment. The algorithm is then modified to make it suitable for a hardware implementation.

The SARSA algorithm involves learning the environment by choosing actions at each state using a policy function. The policy function used in this implementation is that ninety percent of the actions are chosen to maximize reward based on Q-values of the state and ten percent of the actions are taken at random. At each state, an action is chosen and then the environment provides the reward for that action along with the next state. Based on the reward, the Q-values are updated in the Q-table. An episode ends when the agent reaches the destination or detects an obstacle. Until then, the agent explores the environment. By doing this, multiple iterations of this episode allow the agent to identify the optimal path to reach the destination by assigning positive reward for destination and negative reward for obstacles.

Stereo vision is used for the computation of depth based on the binocular disparity between the images of an object in left and right eyes.

The Kinect camera is a motion sensing input device produced by Microsoft, released in 2010. The device generally contains RGB cameras, and infrared projectors and detectors that map depth through structured light and time of flight calculations, which can in turn be used to perform real-time gesture recognition and body skeletal detection, among other capabilities.

Image processing techniques are used to extract and utilize the depth map generated by the Kinect camera.

Figure 2: XBOX 360 Kinect camera

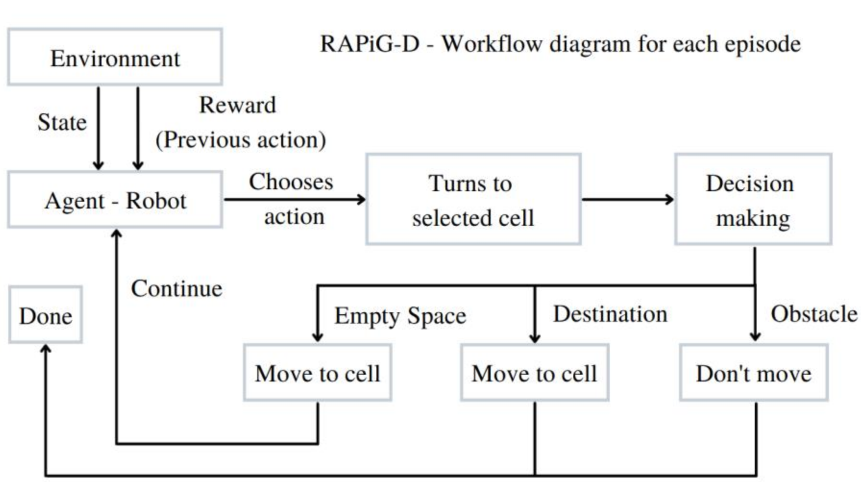

Figure 3: Flow diagram for each episode

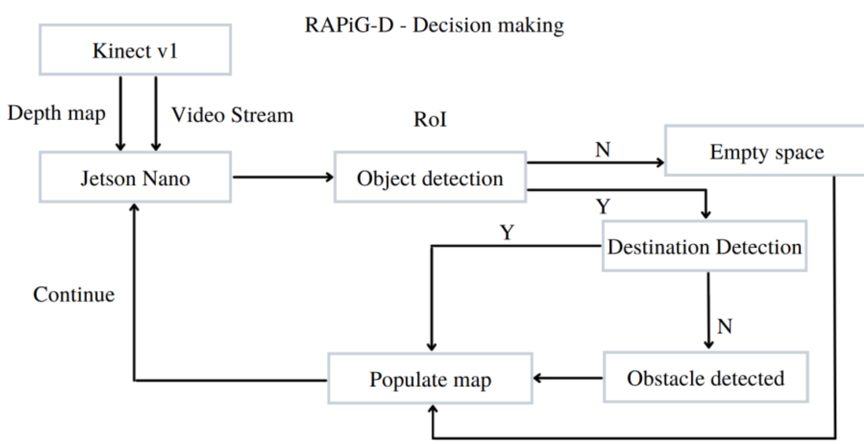

Figure 4: Flow diagram for decision making at each cell

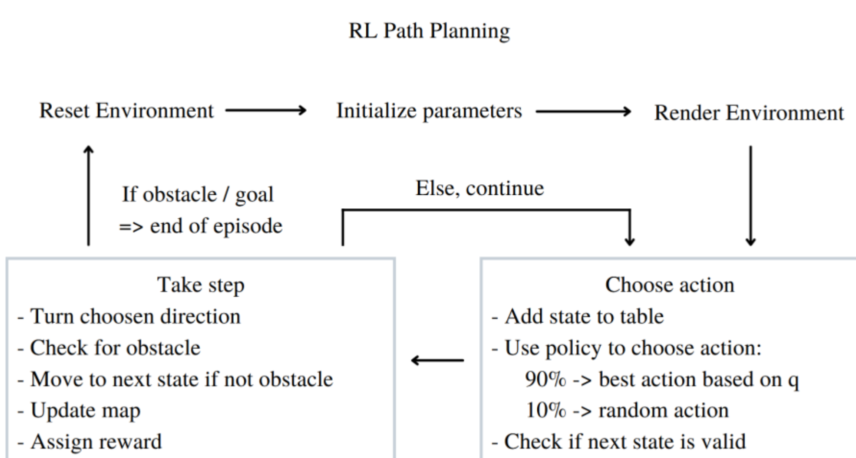

Figure 5: Flow diagram for Path planning algorithm in hardware implementation

To perform hardware implementation, the NVIDIA Jetson Nano Developer Kit is used as the Micro-controller to run the developed algorithm and interface with the XBOX 360 Kinect camera which is shown in Figure 2 and the required sensors for calibrated movement from one cell to the next in the environment. The Kinect camera provides the required depth map by utilizing a stereo camera setup along with an IR camera. An MPU6050 sensor is used to accurately turn by integrating the angular velocity output of the sensor to obtain angular displacement. A Li-Po battery is used to power four 12V 100 RPM DC motors. The Jetson Nano is externally powered using a 5V4A AC adapter.

Figure 6: Front view of the robot designed

Figure 7: Side view of the robot designed

Figure 6 and Figure 7 shows the front and side view of the robot designed for the purpose of implementing the SARSA algorithm as a hardware solution to path planning and to implement obstacle detection using stereo camera-based depth estimation.

Both the software simulation and the hardware implementation follow the same algorithm except for the additional implementation of Obstacle detection using depth estimation in hardware implementation as opposed to feeding obstacle co-ordinates in the simulation.

Figure 3 depicts the workflow of the algorithm implemented for each episode. Figure 4 shows the decision-making process at each cell. Figure 5 shows the algorithm for hardware implementation of path planning. At each state i.e., at each cell the agent – the robot chooses an action and then takes the step if it is valid based on the action by turning to the required direction – up, down, left or right. The step involves checking whether the next state has an obstacle using the depth map generated by stereo camera depth estimation. The obstacle detection is optimized by applying a threshold on the depth map generated to identify objects only in the immediate next cell. Additionally, a region of interest is also applied to detect obstacles accurately. The agent then checks if the next state is an empty space, an obstacle or the destination. This is then used to populate the map of the environment.

A Q-table is initialized with zeros at the start of the first episode which is used to determine how valuable it is to take a particular action at a particular state to maximize reward received. The Q-table is updated according to the reward offered for the next state for each action at the current state. A reward of 1 is offered if the next state is the destination, 0 if it is an empty space and -1 if it is an obstacle.

In each episode, the agent goes around the environment, exploring until it reaches either an obstacle or the destination. If the agent reaches the destination, the path from start point is recorded to check if it is the shortest path. This is used to identify the optimal path to the destination which can then be used to traverse to the destination.

IV. RESULTS AND DISCUSSION

A. Software Implementation

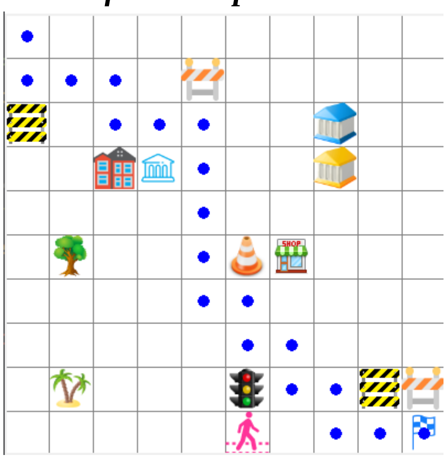

Figure 8: SARSA – Software implementation- Map of environment

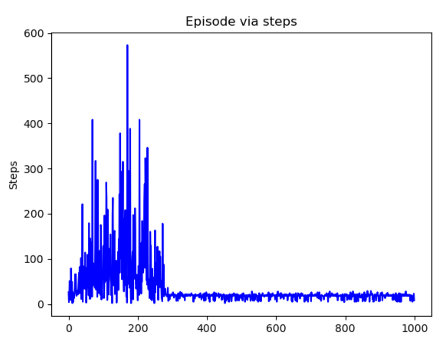

Figure 9: SARSA – Software implementation- Performance metrics

Software implementation was performed on a larger 10x10 environment with a larger number of obstacles and was run for 1000 episodes as software implementation takes less time per episode. Figure 8 shows the map generated by the SARSA algorithm when run as a software simulation and Figure 9 shows the plot of number of steps taken in each episode. The obstacle co-ordinates are pre-determined and fed to the algorithm. The Reinforcement Learning agent moves through environment based on the policy and checks if each co-ordinate is an obstacle, empty space or the destination. By this way, every time it reaches the destination the shortest path is updated allowing the agent to identify the optimal path to the destination while avoiding obstacles.

B. Hardware Implementation

The hardware implementation was then performed using a robot controlled by NVIDIA Jetson Nano with an XBOX 360 Kinect camera for depth estimation in a smaller 3x3 environment with two obstacles. The cell-based environment consists of 2 feet x 2 feet cells.

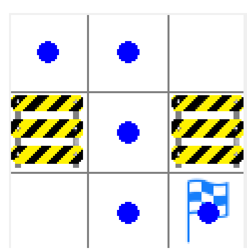

Figure 10: Map generated for 3x3 environment

Figure 10 shows the map generated by the agent as it goes around the environment, exploring and detecting obstacles. The detected obstacles are populated in the map. It also shows the optimal path from start to destination as identified by the agent, thus performing path planning using reinforcement learning.

```
Length of final Q-table = 3
Final Q-table with values from the final route:
              0         1        2     3
[40.0, 0.0]   0.0  0.000008  0.00000  0.00
[40.0, 40.0]  0.0  0.000882 -0.01000 -0.01
[40.0, 80.0]  0.0  0.000000  0.04901  0.00

Length of full Q-table = 8
Full Q-table:
               0         1             2     3
[0.0, 0.0]    0.00 -0.010000  3.615840e-08  0.00
obstacle      0.00  0.000000  0.000000e+00  0.00
[40.0, 0.0]   0.00  0.000008  0.000000e+00  0.00
[80.0, 0.0]   0.00 -0.010000  0.000000e+00  0.00
[40.0, 40.0]  0.00  0.000882 -1.000000e-02 -0.01
[40.0, 80.0]  0.00  0.000000  4.900995e-02  0.00
[0.0, 80.0]  -0.01  0.000000  2.673090e-04  0.00
goal          0.00  0.000000  0.000000e+00  0.00
```

Figure 11: Q-Table for the 3x3 environment – Hardware implementation

Figure 11 shows the Q-table generated by the RL SARSA algorithm which shows how valuable each action is at a given state with the states as rows and the actions as columns – Up, Down, Right, Left.

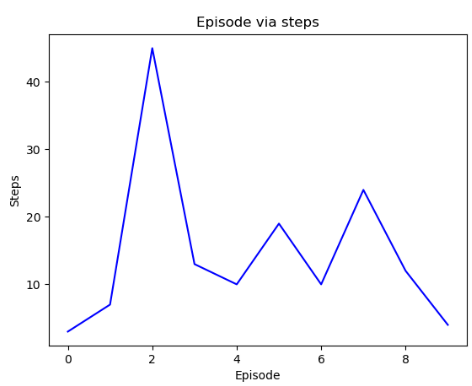

Figure 12: Plot of number of steps in each episode – 3x3 environment

Figure 12 shows the plot of number of steps in each episode. In a 3x3 environment, the agent finds the optimal path within 10 episodes and traverses to the destination in the optimal path based on the Q-table. As the obstacles and the size of the environment increases the time required to identify the optimal path would also increase.

This can be implemented in a garden environment where the robot agent can be tasked to reach a target plant in an optimal path and can monitor or tend to the plant as required.

V. CONCLUSION

Thus, it is seen that the robot agent successfully performs Reinforcement Learning based path planning to accurately map the environment and its obstacles and identify the optimal path to reach a given destination. In the 3x3 environment, the robot performs accurate obstacle detection using depth estimation and by employing the SARSA algorithm, finds the optimal path to the destination within 10 episodes.

Therefore, SARSA based path planning has been tested using hardware implementation and its ability to determine the optimal path to reach a destination in an unknown environment while detecting and avoiding obstacles has been validated.

VI. FUTURE WORKS

This implementation can be extended to environments larger environments that are more dynamic. Future works may involve increasing the degrees of freedom of the environment and going beyond cell-based environments.

It can have diversified applications ranging from garden or warehouse management, serving food at restaurants, Air crash investigations, Search and Rescue operations and even Space Exploration.

VI. REFERENCES

[1] S. Zheng and H. Liu, "Improved Multi-Agent Deep Deterministic Policy Gradient for Path Planning-Based Crowd Simulation" in IEEE Access, vol. 7, pp. 147755-147770, 2019.

[2] Xinyuan Zhou, Peng Wu, Haifeng Zhang, Weihong Guo and Yuanchang Liu, "Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning", IEEE Access, vol. 7, 2019

[3] Chen Chen, Jiange Jiang, Ning Lv and Siyu Li, "An Intelligent Path Planning Scheme of Autonomous Vehicles Platoon Using Deep Reinforcement Learning on Network Edge", IEEE Access, vol. 8, 2020.

[4] Chao Wang, Jian Wang, SeYuan Shen, and Xudong Zhang," Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach", IEEE Transactions On Vehicular Technology, vol. 68, pp. 2124-2136, March 2019.

[5] D. Ebrahimi, S. Sharafeddine, P. -H. Ho, and C. Assi, "Autonomous UAV Trajectory for Localizing Ground Objects: A Reinforcement Learning Approach," in IEEE Transactions on Mobile Computing, vol. 20, pp. 1312-1324, 1 April 2021.

[6] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile Robot Path Planning in Dynamic Environments Through Globally Guided Reinforcement Learning," in IEEE Robotics and Automation Letters, vol. 5, pp. 6932-6939, Oct. 2020.

[7] A. Konar, I. Goswami Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 43, pp. 1141-1153, Sept. 2013

[8] J. Xin, H. Zhao, D. Liu, and M. Li, "Application of deep reinforcement learning in mobile robot path planning," In: Chinese Automation Congress (CAC), 2017.

[9] V. N. Sichkar, "Reinforcement Learning Algorithms in Global Path Planning for Mobile Robot," In: International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2019, pp. 1-5

[10] Valentyn N. Sichkar, "Reinforcement Learning Algorithms in Global Path Planning for Mobile Robot," In: International Conference on Industrial Engineering, Applications and Manufacturing, 2019

[11] P. Gao, Z. Liu, Z. Wu, and D. Wang, "A Global Path Planning Algorithm for Robots Using Reinforcement Learning," In: IEEE International Conference on Robotics and Biomimetics (ROBIO), 2019.

[12] Y. Long, and H. He, "Robot path planning based on deep reinforcement learning," I*n:* IEEE Conference on Telecommunications, Optics and Computer Science (TOCS), 2020

[13] P. Mohan, L. Sharma and P. Narayan, "Optimal Path Finding using Iterative SARSA," In: 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 811-817.