Journal of Trends in Computer Science and Smart Technology (ISSN: 2582-4104)
www.irojournals.com/tcsst/

Path Planning of Mobile Robot Using Reinforcement Learning

Kiran G Krishnan[1*], Abhishek Mohan[2], S. Vishnu[3], Steve Abraham Eapen[4], Amith Raj[5], Jeevamma Jacob[6]

[1-5]National Institute of Technology Calicut, Kozhikode, India
[6]Professor, National Institute of Technology Calicut, Kozhikode, India

E-mail: *kichukiran97@gmail.com

Abstract

In complex planning and control operations and tasks like manipulating objects, assisting experts in various fields, navigating outdoor environments, and exploring uncharted territory, modern robots are designed to complement or completely replace humans. Even for those skilled in robot programming, designing a control schema for such robots to carry out these tasks is typically a challenging process that necessitates starting from scratch with a new and distinct controller for each task. The designer must consider the wide range of circumstances the robot might encounter. This kind of manual programming is typically expensive and time consuming. It would be more beneficial if a robot could learn the task on its own rather than having to be preprogrammed to perform all these tasks. In this paper, a method for the path planning of a robot in a known environment is implemented using Q-Learning by finding an optimal path from a specified starting and ending point.

Keywords: Path planning, reinforcement learning, Q-learning, mobile robot, robot operating system

1. Introduction

A completely autonomous robot has the ability to learn about its surroundings, work for a lengthy amount of time without assistance from humans, move all or part of itself around its operational environment on its own, and avoid situations that could be dangerous to people or property. An autonomous robot can also pick up new skills or information, adapting to new ways of doing things or to changing environments. So, by using a static algorithm like A*, Dijkstra or D* algorithm, doesn't necessarily accomplish the tasks even though they are faster in path planning.

DOI: https://doi.org/10.36548/jtcsst.2022.3.004
Received: 11.07.2022, received in revised form: 05.08.2022, accepted: 19.08.2022, published: 29.08.2022

The navigation of mobile robots generally consists of the following six interrelated skills such as perception, exploration, mapping, localization, path planning and path execution:

- Perception is the act of obtaining and interpreting information from the senses.
- Exploration refers to the tactic utilized by the robot in the process of determining the subsequent path to take.
- Mapping is the process of constructing a representation of space or a model of an environment by making use of the sensory data that is taken in.
- Localization refers to the process of simultaneously estimating where on the spatial map the robot is located while it is under the control of its navigation system.
- Path planning refers to the process of determining whether a particular route towards a desired destination is the most effective one.
- Path execution refers to the process of determining and adapting motor actions to changes in the environment, including the avoidance of obstacles.

When it comes to the navigation of a robot, the most important factor is the planning of its path. Because there is a greater possibility of agents colliding with one another in a multi-agent system, the significance of route planning is multiplied many times over. The topic of discussion in this article is a novel approach to route planning that makes use of reinforcement learning. In addition to the innovative algorithm for route planning, an innovative approach to communication is also being utilized in the process of putting this algorithm into practice.

To train machine learning models, reinforcement learning is used, and the models are then asked to make a series of decisions. The agent develops the ability to complete a task despite the potentially difficulty and unpredictability of the surrounding environment. During the process of reinforcement learning, an artificial intelligence may run into a situation that is analogous to a game. The computer will use a process of trial and error in order to figure out how to solve the problem. The actions that the artificial intelligence takes to get the machine to do what the programmer wants are evaluated and either rewarded or punished by the system. Its purpose is to achieve a total reward that is as high as possible. In this instance of path planning, a Q-learning algorithm has been used in order to successfully complete the

task. Using the model-free reinforcement learning algorithm known as Q-learning, an agent can learn a policy that specifies what action to take and under what conditions it should take that action. Since it does not require a model of the environment (hence the term "model-free"), it is able to deal with problems involving stochastic transitions and rewards without the necessity of making any adjustments [1]. Figure 1 shows the reinforcement learning model.

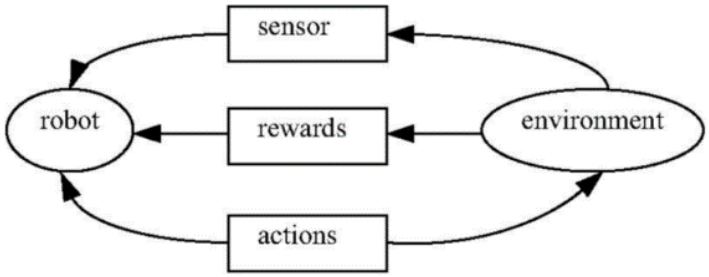

Figure 1. Reinforcement Learning Model

Mobile robots will acquire the capability to adapt to an ever-changing environment once reinforcement learning is implemented into the path planning process. As a result, these robots will be able to complete their tasks independent of any assistance from a human being.

In this paper, a reinforcement learning algorithm has been implemented for the path planning of a mobile robot by considering a known environment. The algorithm has been simulated using gazebo and implemented using fire bird V robot also.

This paper is organized as follows: Section II explains the simulations done using python, gazebo and OpenAI gym. The overview of the system has been explained in section III. The results and conclusion drawn from the project has mentioned in sections IV and V respectively.

2. Simulations

Python, gazebo, and OpenAI gym are used in the simulations that are run. The Tkinter library is used to create a basic setting, and the RL algorithm is implemented in order to determine the most efficient route between the starting point and the end state. Grids have been placed throughout the environment. Therefore, the path is the process of moving through the grid's cells in order. The captured image is presented below in Figure 2.

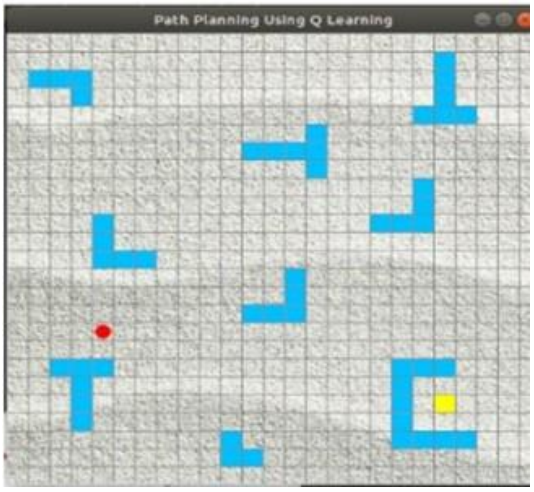

Figure 2. Python Simulation

The use of gazebo and OpenAI gym allows for more accurate simulations to be run. Gazebo is a 3D simulation software that allows for the importation of almost all conceivable physical features. Gazebo is responsible for the creation and distribution of a model robot known as Turtle bot. For the purpose of simulation, an environment has been created within the simulator. OpenAI is a library that enables to construct environments quickly and easily and to efficiently implement RL algorithms. In this case, the RL algorithm has been used to locate the optimal path. Because the path is unbroken, one must go through each of the possible outcomes. Therefore, it took a longer period to finish. The captured image from the simulation can be found in Figure 3.

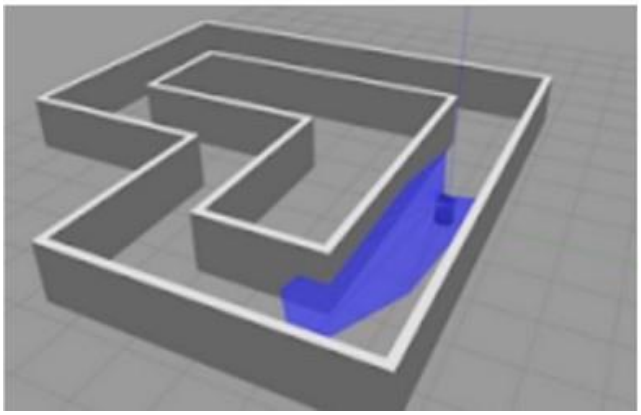

Figure 3. Gazebo Simulation

3. System Overview

In order to implement and test the algorithm, a hardware system is required. Fire Bird V robot, made by Nex robotics has been used as the hardware platform. It has an integrated

ATmega2560 and Zigbee module on it. A model of Fire Bird V and its internal components are shown in Figure 4 and Figure 5 respectively.

Figure 4. Fire Bird V

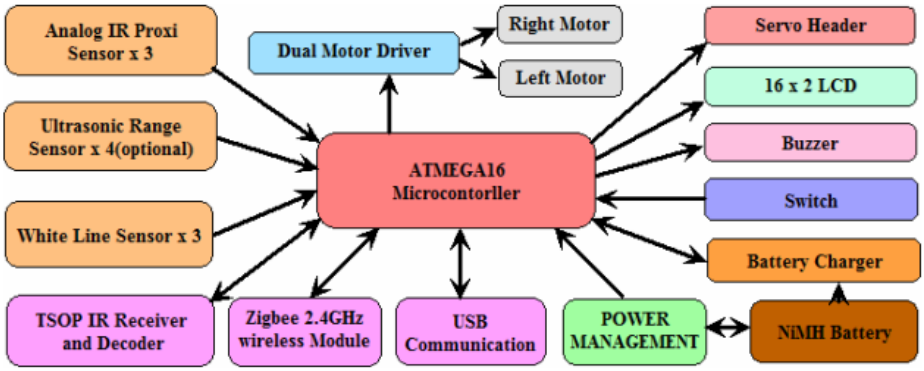

Figure 5. Internal Components of Fire Bird V

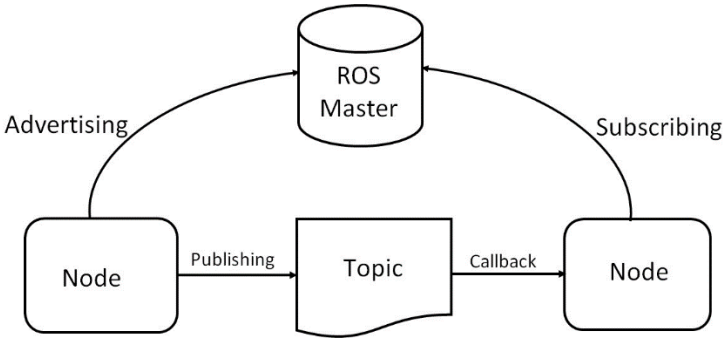

Figure 6. ROS Workflow

In the software side, Robot Operating System (ROS), OpenCV, OpenAI gym and Gazebo are used. ROS has been used as a middleware for facilitating the communication between the nodes. OpenCV is an image processing tool which has been used for capturing

the environment through the camera feed. OpenAI gym and Gazebo have been used for the simulation purposes [2]. Figure 6 and Figure 7 show the ROS workflow and functionalities of OpenCV libraries respectively.

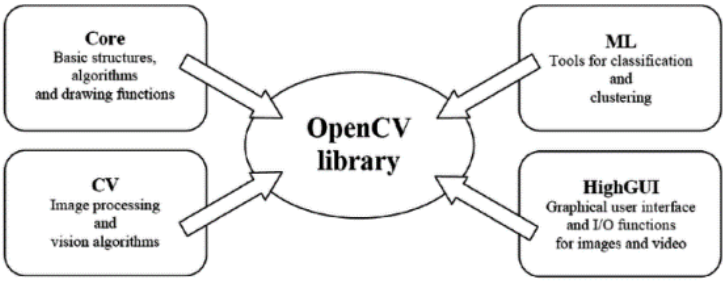

Figure 7. OpenCV Functionalities

3.1 Working Outline

For the path planning of a robot, environment should be known to the system [3]. This is achieved through an overhead camera. Using the overhead camera, obstacles can be isolated from the surroundings using image processing by OpenCV. ROS is used as the middleware [4] [5]. After isolating the obstacles from the surroundings, the grid which has the obstacles will have the value 255 whereas, free grids will have the value 0. In this matrix, Reinforcement Learning Algorithm is applied for getting the optimal path. The path generated is shown in Figure 8. After getting the path, the bot follows that path to reach the goal state.

Figure 8. Path generated from the algorithm

3.2 Algorithm

The maze is considered to be a two-dimensional matrix of size 12x12. Each cell of the matrix represents a state that the agent can transition to. The agent can only take 4 actions from a state [6][7]. A Q-value is assigned to all actions at each state [8], and stored in a Q-

Matrix of size 12x12x4 (three-dimensional). The agent starts at the cell with indices (1,1) and the goal state is the cell with indices (7,11). The walls of the maze are set as 1, and paths are set as 0. The endpoint of the maze is set as 2. A sample maze has been shown in Figure 9.

1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	0	0	0	0	1
1	0	1	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	0	0	1	1
1	0	1	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	1	0	0	1	1
1	0	1	1	0	0	0	1	0	1	1	1
1	1	1	1	0	1	1	1	0	1	1	1
1	1	1	1	0	0	0	1	0	1	1	1
1	1	1	1	1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	2	1	1	1	1

Figure 9. Sample Maze

At each cell, the agent decides which action to take based on the policy (-greedy). Number generates values from 0 to 101. If the number is less, a random action is chosen. If the number is greater, the action with maximum Q-value is chosen. If the action is valid, the agent gets rewarded, and the Q-value is updated as in Eq. (1). Agent also transitions to the next state. Else, the agent is punished, and the Q-value is updated as in Eq. (1).

$$q[y][x][p] = q[y][x][p] + a*\{rwd(maze,x,y-1) + g*fMV(q,y-1,x) - q[y][x][p] \quad (1)$$

- q[y][x][p] is the Q-value of the state-action pair ((x,y),p).
- x and y denote the indices of the current maze cell the agent is in.
- a is the learning rate set at 0.5, and g is the discount factor set at 0.8
- The function fMV() finds the value of maximum q for the state (x,y-1) which the agent transitions to.

The function rwd() determines the reward for the transition. The reward for a valid state transition is set as 0. If the agent reaches the goal state, a reward of +10 is given to it. For an invalid action, the agent is given a punishment of -10.

Agent starts taking a huge number of iterations before it can reach the terminal state [9] [10]. After a few episodes, the number of state transitions required before the agent reaches the terminal state reduces drastically. By around 25 episodes, the algorithm

converges to the solution. After the training is completed, the agent traverses through the maze, and can solve the maze in 22 steps. The result has been shown in Figure 10.

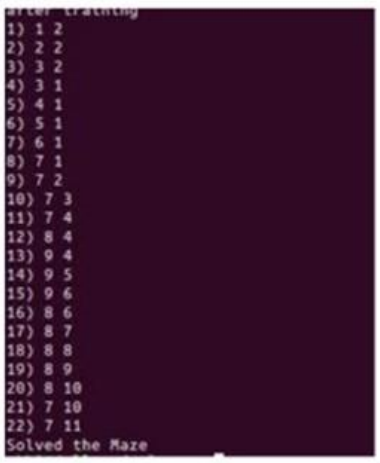

Figure 10. Result after the iterations

4. Results and Discussion

The development of an algorithm for the path planning of mobile robots has been accomplished with the help of Q-learning. Gazebo and OpenAI gym have been used to simulate it, and the results have been positive. The turtle bot has been successful in finding a path that is both efficient and effective through the environment. For most of the time, the path is converged quickly. But it is observed that the amount of time needed to converge the path increases proportionally with the level of complexity.

Figure 11. Implementation using Fire Bird V

Following the simulation, the algorithm is successfully put into action by making use of a fire bird V robot, as can be seen in Figure 11. It has been discovered that the system is able to quickly adjust to the shifting environment. However, as a result of the increasing complexity of the environment, the amount of time required to converge also increases. On

the other hand, thanks to this algorithm, there is now access to a fresh avenue for additional research and study that can be carried out in the future in order to speed up the convergence process.

5. Conclusion and Future work

The integration of the Robot Operating System (ROS), which is a novel method for communication between the various components of a system, contributes to the acceleration and enhancement of operational efficiencies. In addition, ROS features provisions simplify the process of integrating multiple agents. The system is able to take into account the dynamic changes in the surrounding environment as it uses reinforcement learning algorithm. When the Reinforcement Learning algorithm is optimized, the system is able to converge more quickly on the best path to take. The results obtained from the performed simulations show that the path obtained from RL algorithms is more optimally optimized than the path obtained from other search algorithms such as A* algorithms, BFS algorithms, and so on. These algorithms are adaptable enough to be used in environments that are unknown. Because of this, specialized mapping agents ought to be maintained for the purpose of mapping the environment. Following the execution of this algorithm, additional robots can be incorporated into the system. At the moment, the convergence of the result is extremely sluggish. In order to increase the speed of convergence, the RL algorithm needs to be optimized and improved. In addition to this, additional sensors can be incorporated in order to improve the effectiveness and efficiency of the perception and processing of the environment. It is possible to implement learning strategies that are more effective for use in path planning and other applications.

References

[1] Ee Soong Low, Pauline Ong, Kah Chun Cheah, “Solving the optimal path planning of a mobile robot using improved Q-Learning”, International Journal for Robotics and Autonomous System(Elsevier), vol. 115, pp. 160-169, 2019.

[2] Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, Alejandro Hernandez Cordero,”Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo”. Erle Robotics, pp. 322-340, 2017.

[3] Khaled Alaa, Nicolo Botteghi, Beril Sirmacek, Mannes Poel,”Towards continuous control for mobile robot navigation: A Reinforcement Learning And SLAM based approach”, International Control Conference, pp. 932-940. May 2019.

[4] Murat Koseoglu, Orkan Murat Celik, Omer Pektas, "Design of an autonomous mobile robot based on ROS", International Artificial Intelligence and Data Processing Symposium(IDAP), pp. 1024-1030, 2017.

[5] Lei Tai, Giuseppe Paolo, Ming Liu, "Virtual-to-real Deep Reinforcement Learning: continuous control of mobile robots for mapless navigation",IEEE Transaction on Robotics and Automation, vol 35, pp. 799-816, 2019.

[6] Yu Fan Chen, Michael Everett, Miao Liu, Jonathan P How, "Socially aware motion planning with Deep Reinforcement Learning", Cornell University Thesis, pp. 120-135, 2017

[7] G. Priyandoko, T Y Ming, M S H Achmad, "Mapping of unknown industrial plant using ROS-based navigation mobile robot", International Conference on Computer Engineering and Science, pp. 767-772, May 2018.

[8] Zhiqiang Tang, Peiyi Wang, Wenci Xin, and Cecilia Laschi. Learning-based approach for a soft assistive robotic arm to achieve simultaneous position and force control. IEEE Robotics and Automation Letters, 2022.

[9] Krishnan, K. G. (2022). Using Deep Reinforcement Learning For Robot Arm Control. Journal of Artificial Intelligence and Capsule Networks, 4(3), 160-166. doi:10.36548/jaicn.2022.3.002

[10] Baoguo Xu, Wenlong Li, Deping Liu, Kun Zhang, Minmin Miao, Guozheng Xu, and Aiguo Song. Continuous hybrid bci control for robotic arm using noninvasive electroencephalogram, computer vision, and eye tracking. Mathematics, 10(4):618, 2022.