Reinforcement Learning for Mobile Robot's Environment Exploration

Reinforcement Learning for Mobile Robot's Environment Exploration

Sean W H Teoh[1], Kamarulzaman Kamarudin[1,2], Nasr A N Ali[1], Muhammad M M Zainal[1], Mohd R Manan[1], Syed M Mamduh[2]

[1] Faculty of Electrical Engineering Technology, Universiti Malaysia Perlis, Kampus Pauh Putra, 02600 Arau, Perlis, Malaysia
[2] Centre of Excellence for Advanced Sensor Technology (CEASTech), Universiti Malaysia Perlis, Arau 02600, Perlis, Malaysia.

E-mail: kamarulzaman@unimap.edu.my

Abstract. Mobile robots are being are being applied in various industries to perform repetitive or dangerous tasks for humans to carry out. Autonomous mobile robots are more capable than automated guided vehicles (AGV) due to their ability to be adaptable to their environment which is important for exploration of unknown environments. It is difficult to program autonomous mobile robots to adapt to various situations it may face, thus machine learning can be applied to allow a mobile robot to learn how to navigate through environments by itself. Reinforcement learning is applied in this project so that a differential drive mobile robot can learn how to navigate through its environment while avoiding collision with surrounding walls and obstacles. The reinforcement learning process is simulated by using the Robot Operating System (ROS) and its simulator Gazebo. Controlled simulation environments are created using Gazebo for the purposes of training and performance testing. Simultaneous Localization and Mapping (SLAM) will be applied to generate a map of the environment. At the end of this project, the Turtlebot3 is able to map smaller controlled environments ranging between 18m^2 to 27m^2 without colliding with the surrounding walls.

1. Introduction

Robots are increasingly being applied to various aspects of human lives, from waiter robots in restaurants [1] to numerous lifting robots in warehouses [2], where robots assist humans to carry out repetitive tasks. Under most circumstances, robots can be pre-programmed to carry out the associated task with a higher efficiency than their human counterparts under optimal and controlled environments [3], however in reality there are many external forces or conditions that could affect the performance of the robot. For example, facing unforeseen obstacles or simple repositioning [4] which would be of little issue to a human but could completely stop some robots from performing their task without human aid. To overcome this issue, autonomy of robots has become one of the most popular fields of study in recent years [5]. An autonomous mobile robot means it is capable of performing a task that involves some decision making without human supervision. An important aspect of autonomous mobile robots is the

ability to move from one point to another safely and efficiently, this is due to the numerous approaches that can be taken and the robot has to make the decision itself, unlike automated guided vehicles which have predetermined paths usually defined by special tape [6]. This issue is known as path planning and the mobile robot requires information about its current location, surrounding environment and target location before it can select an action, this is referred to as localization and mapping [7]. Rather than program the mobile robot for numerous possibilities it could face, machine learning can be applied so that the mobile robot can learn to interact with the given environment by itself, this is called reinforcement learning [8]. Reinforcement learning is suitable for this application because the mobile robot is an agent that is interacting with its environment and learns which actions are favorable through positive or negative feedback designated by a reward system.

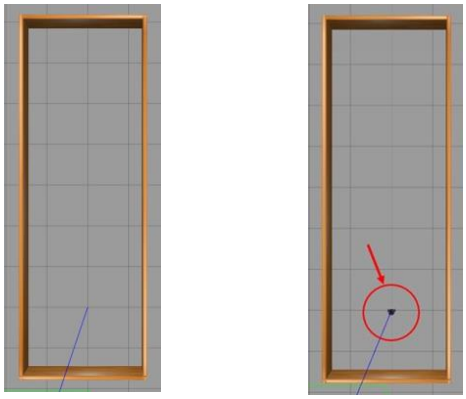
2. Methodology

2.1. Simulation Environment

ROS is an open-source program, and it was selected as to focus on the software rather than the hardware aspects of mobile robots. An openly available simulated differential drive mobile robot is also needed for this project. The Turtlebot3 simulated mobile robot model shown in Figure 1 was chosen for this project as it can be used with ROS Noetic as well as being a small differential drive mobile robot that comes with its own Lidar sensor and a SLAM program, g-mapping to be specific.

Figure 1. Turtlebot3

Environments are also needed, through the use of the building editor tool available in Gazebo simulator, simple training environments are created, consisting of walls with 1.0m in height and 0.1m thickness with varying lengths ranging from 2.0m to 10.0m to create closed environments for the Turtlebot3 to train in. Training environments known as “worlds” are created with the goal of the Turtlebot3 to learn how to take specific motions, step-by-step, an example of this is a rectangular world that is 3.0m x 8.0m in size in this project referred to as Environment-A is used to train the Turtlebot3 to move forward without hitting the walls as shown in Figure 2 (a). The starting position of the mobile robot is shown in each figure of each environment and is circled in red such as in Figure 2 (b). The Turtlebot3 is trained to carry out specific motions in each environment, thus allowing the mobile robot to learn basic motions one followed by another before trying to handle more complicated environments which require the execution of multiple different motions to traverse without collision. The mobile robot will learn how to move in the structured order of (Forward > Turn Left > Turn Right > Turn Back > Turn Sharp Left > Turn Sharp Right > Multiple Action). This learning structure ensures that the mobile robot has properly learned how to carry out simple motions in fixed environments before carrying out more complex combinations of motions to traverse more complicated environments.



(a) (b)

Figure 2. (a) Environment-A and
(b) Turtlebot3 Starting Position in Red Circle

2.2. SLAM implementation

Once the environments have been set-up, Simultaneous Localization and Mapping (SLAM) is used to generate maps from the data received from the 2D-Lidar sensor on the Turtlebot3. Since the Turtlebot3 has a 2D-Lidar Sensor, the maps generated will also be 2 dimensional or 2D as well. Mapping is one of the key aspects of exploration as it is one of the primary outputs desired. The specific SLAM algorithm used in this project is called g-mapping and comes with the Turtlebot3 simulated model. The Turtlebot3 will be manually driven across the environments to see what the output of a completed map of the environment should look as well as if the g-mapping algorithm has any limitations when generating maps in the different kinds of environments as different SLAM algorithms have their own strengths and weaknesses. This mapping will be carried out via the "Teleop Function" which stands for teleoperation through a laptop keyboard.

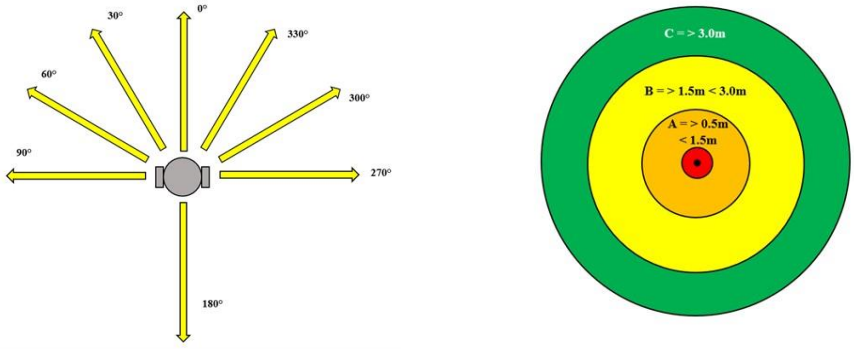
2.3. Reinforcement Learning in Gazebo Simulation

Based on previous literature, mobile robot environment exploration and mapping can be achieved through the use of reinforcement learning, but exploring unknown environments is more complex than navigating to a goal point and suffers from the issue of dimensionality [9]. Deep reinforcement learning or reinforcement learning with rapidly-exploring random tree are methods that have been successfully used for environment exploration but is complex in implementation [9-10]. Thus, this project aims to achieve simplified environment exploration.

The simulation environments in Gazebo have varying sizes, thus implementing a coordinate- based system would be less efficient given larger environments. Besides that, a mobile robot does not teleport from one state to another, the robot actually has to move between states, resulting in constantly changing orientations. Lastly, the mobile robot is actually relying on its 2D-Lidar to perceive its surroundings and does not know what the entire map looks like, making motion through a coordinate system difficult. These factors result in the coordinate-based state system being replaced with a 2D-Lidar distance-based state system. A select number of Lidar output angles are divided into fixed distances, depending on the distance of the closest wall or obstacle within range, the system will return a corresponding alphabet. The combination of all selected Lidar angle readings will be used as states for reinforcement learning in this project. This results in having a fixed number of states regardless of what kind of environment it is applied in as well as not needing knowledge of the overall map or specific coordinate location.

8 2D-Lidar output angles have been selected to be used for state identification. 7 out of the 8 selected angles are located in the forward 180° of the mobile robot with an equal angle difference of 30° between each selected 2D-Lidar output angle. Each output angle will be divided alphabetically according to the 4 possible ranges. The default maximum distance measurements of the 2D-Lidar of the Turtlebot3 is around 3.2m before returning (inf) as a reading, thus readings equal to or above 3.0m is labelled as (C)

which is considered clear in the given direction as there is no obstacle or wall within range to be considered, the next range of values will be between 1.5m and 3.0m labelled as (B) which is considered as having an obstacle or wall within range but still a safe distance away, followed by values between 0.5m and 1.5 labelled as (A) which is considered as having an obstacle or wall within range that is close to the Turtlebot3. Lastly if there is any obstacle or wall within 0.5m of the mobile robot, it is labelled as (e) which is considered dangerously close and is the negative terminal condition. The 2D-Lidar output angles and the preset ranges are shown in Figure 3. The total possible number states for this system are equal to (3^8 + 1) which is 6562. This may seem like a large number at first, but this number is fixed regardless of environment unlike the coordinate system which has a varying number of states depending on environment size.



(a) (b)

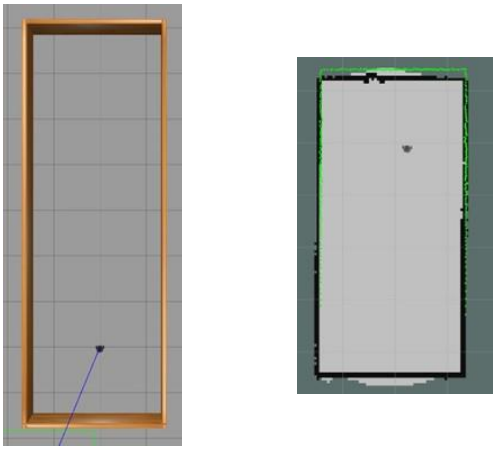
Figure 3. (a) 2D-Lidar Output Angles and (b) Ranges for Reinforcement Learning States

For the actions, the Turtlebot3 has been set to only be able to carry out 6 possible actions (turn left, slightly left, forward, slightly right, turn right, stationary rotation). This is more than enough to navigate through a controlled environment. Due to the goal of exploration, a traditional goal point is not applicable. Thus, to assist in the process of reward shaping without a goal point, the mobile robot will have the goal of following the left wall and the rewards for reinforcement learning will be representative of this goal. The states experienced by the Turtlebot3 and its corresponding reward values are processed through a custom node in ROS that is able to convert the 2D-Lidar information to states and corresponding rewards as it receives information from the topic called "/scan" that publishes the 2D-Lidar data of the Turtlebot3, the information processed by the custom node is then passed to another custom node that handles reinforcement learning. Since there are (6561 + 1) possible states and 6 possible actions, the maximum total data required is (65616) + 1which is 39367 which will be stored in a .json file which can store the python dictionary for future application in other environments so that learning can be continuous across multiple environments.

3. Results

3.1. G-mapping SLAM in Different Environments

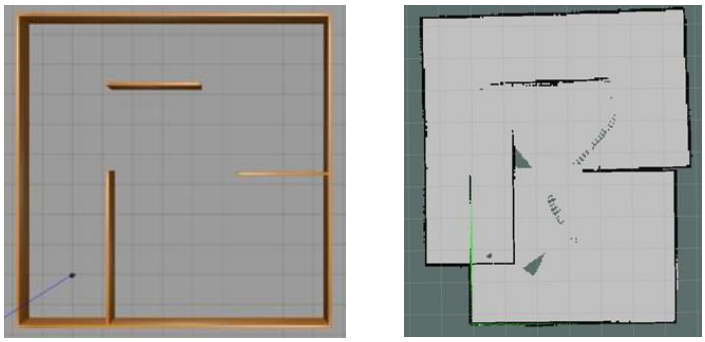
The g-mapping SLAM algorithm cannot estimate the length of a wall well probably due to a long straight corridor not having other features to compare to. This is displayed in Figure 4 (b). G-mapping applies scan-matching which compares multiple scans to find matching features to combine the scans together in the correct orientation. Thus, a larger map with little to no features may result in inaccurate maps being generated.



(a) (b)

Figure 4. (a) Environment-A and (b) Map of Environment-A

However, due to the large size of Environment-I, and having features that are spread out far enough to not be present in every scan, the map produced is increasingly inaccurate as the Turtlebot3 traverses across the environment. This is shown in Figure 5 (b). Thus, g-mapping SLAM is applicable but may not be the best SLAM algorithm for mapping in this environment and other algorithms can be considered for future improvements such as Hector-SLAM or Google Cartographer.



(a) (b)

Figure 5. (a) Environment-I and (b) Map of Environment-I

3.2. Parameter Analysis for Reinforcement Learning

There are many variables that can affect the reinforcement learning process and overall output which include but are not limited to the learning rate α which affects the speed at which convergence is achieved and discount rate γ which determines how important future rewards are to the agent. Thus, to analyze the effect of these 2 variables on the overall performance of the Turtlebot3, experiments were carried out to determine which values of α and γ result in the best performance in the Turtlebot3 by observing the time taken for the Turtlebot3 to traverse Environment-A and generate a complete map. Time taken is recorded using a stopwatch application on a smartphone. When different values of α are tested, γ is fixed at 0.5, and similarly when different values of γ are tested, α is fixed at 0.5. All other variables remain the same, such as episode count at 1500 and the probability of random actions epsilon ε.

Table 1. Mapping time for Environment-A for Variable α

Variable α	Time (s)
0.2	35.64
0.3	37.13
0.4	38.59
0.5	40.27
0.6	38.25
0.7	39.18
0.8	47.71

Table 2. Mapping time for Environment-A for Variable γ

Variable γ	Time (s)
0.2	38.66
0.3	39.02
0.4	38.34
0.5	40.27
0.6	39.43
0.7	38.97
0.8	36.55

From Table 1 and Table 2, it is shown that generally the lower the value of α, the better the overall performance in terms of time to map an environment and the higher the value of γ the better the overall performance in terms of time to map an environment. Thus, these values of α and γ will be used in training going forward to maximize overall performance. Besides the values of variables used in reinforcement learning, another factor that can affect the performance of the mobile robot in exploration is the order of the type of environment used during training and the speed at which environment complexity is increased. Due to the fact that there are 39367 possible combinations of state and actions for the mobile robot to experience, unstructured learning will require very long hours of learning and will lead to irregular behaviour. This problem is present in larger environments where the 2-D Lidar output can return many more possible combinations of ranges compared to a smaller environment.

3.3. Repeatability Test

The last experiment will be to test if the trained robot is able to exploit the knowledge it has gained through training in a repeatable manor, a mobile robot that has an unpredictable pattern of motion under similar conditions is undesirable as it may carry out actions that are unintended. It is also important that the mobile robot is able to fully exploit its knowledge especially during application in a real environment as mobile robots shouldn't carry out actions that could lead it to dangerous situations to prevent damaging itself. For this experiment, the trained Turtlebot3 is tested in 3 environments, Environment-A for basic motion testing, Environment-G for multiple action testing and Environment-I for complex environment testing.

The Turtlebot3 is able to navigate completely through Environment-A without colliding with surrounding walls and is able to generate a map consistently as proven in Table 3. The Turtlebot3 is unable to fully map environment-G and environment-I however the path taken is consistent.

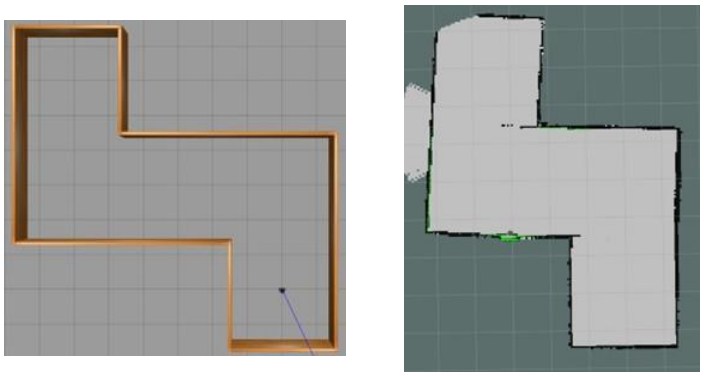
The Turtlebot3 is able to traverse through smaller environments of surface area $18m^2$ and $27m^2$ and generated complete maps. The Turtlebot3 is able to traverse partially through an environment of $45m^2$ in size and generated a partially complete map as depicted in Figure 6 (b). Lastly, the Turtlebot3 fails to traverse the environment of $100m^2$ in size and crashes into a corner before it can complete the mapping process. This is shown in Table 4.

Table 3. Mapping time for Environment-A

No	Time (s)
1	40.27
2	40.79
3	40.81
Average	40.62

Table 4. Performance Across Environments of Different Sizes

No	Area of Environment (m^2)	Average Time (s)	Map Completion
1	18	40.62	Complete
2	27	44.74	Complete
3	45	92.70	Partially Complete
4	100	-	Incomplete



(a) (b)

Figure 6. (a) Environment-G and (b) Map of Environment-G

Thus, it can be concluded that the mobile robot is able to learn and apply its knowledge through reinforcement learning in a repeatable manner however as the size and complexity of the environment increases, the Turtlebot3 is less capable of traversing the environment, this is due to insufficient training to be able to handle every possible state of which there are 39367 as the Turtlebot3 was only trained in smaller confined spaces. Another issue could be the reward shaping as it is observed that the Turtlebot3 has trouble turning right as the reward distribution make the Turtlebot3 favour the left. Thus, more tuning of variables such as rewards, α and γ can yield better performance.

4. Conclusion

In conclusion, Gazebo ROS is an effective tool which can assist robot developers in simulating robots and environments for the experimentation or training of robots without the need to own an actual robot or construct a real environment. Through the use of the building editor tool in Gazebo, the objective of creating simulation environments for mobile robot environment exploration is achieved. Simultaneous Localization and Mapping (SLAM) is an essential tool in exploration to generate a map of the environment. G-mapping can produce an accurate map of the environment given ideal situations but other algorithms can be considered as G-mapping has limitations which make it situational depending on the features of the environment. Thus, the objective of implementing SLAM algorithm for mapping is achieved. At the end of this project, reinforcement learning was successfully applied to a mobile robot to learn how to traverse different environments mostly between 18m^2 and 25m^2 of surface area however

more training is required so that the mobile robot can approach many more kinds of environments without failure. The objective of developing a reinforcement learning framework for map exploration is achieved but with much room for improvement. The overall project can be improved given more time to improve the training speed and experimenting with alternative SLAM algorithms and further tuning of variables in the reinforcement learning process.

Acknowledgement

The authors acknowledge the financial support provided by the Ministry of Higher Education through the Fundamental Research Grant Scheme (FRGS) under a grant number of FRGS/1/2022/TK08/UNIMAP/03/13.

References

[1] Cheong A, Foo E, Lau M, Chen J, and Gan H 2015 Development of a Robotics Waiter System for the food and beverage industry *The Third Intl. Conf. on Advances in Mechanical and Robotics Engineering*
[2] Vivaldini K C T, Galdames J P M, Becker M, and Caurin G A P 2009 Automatic Routing of Forklift Robots in Warehouse Applications *20th international conference mechanical engineering*
[3] Barosz P, Golda G, and Kampa A 2020 Efficiency Analysis of Manufacturing Line with Industrial Robots and Human Operators *Applied Sciences*
[4] Majdik A, Popa M, Tamas L, Szoke I and Lazea G 2010 New approach in solving the kidnapped robot problem *6th German Conference on Robotics*
[5] Siegwart R, Nourbakhsh I R, and Scaramuzza D 2011 *Introduction to Autonomous Mobile Robots, second edition* (USA: MIT Press) p 472
[6] Fragapane G, De Koster R, Sgarbossa F and Strandhagen J O 2021 Planning and control of autonomous mobile robots for intralogistics *European Journal of Operational Research*
[7] Ab Wahab M N, Lee C M, Akbar M F and Hassan F H 2020 Path Planning for Mobile Robot Navigation in Unknown Indoor Environments Using Hybrid PSOFS Algorithm *IEEE Access*
[8] Sutton R S and Barto A G 2018 *Reinforcement Learning, second edition: An Introduction* (USA: MIT Press) p 552
[9] Niroui F, Zhang K, Kashino Z and Nejat G 2019 Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments *IEEE Robotics and Automation Letters*
[10] Chiang H T L, Hsu J, Fiser M, Tapia L and Faust A 2019 RL-RRT: Kinodynamic Motion Planning via Learning Reachability Estimators From RL Policies *IEEE Robotics and Automation Letters*