Reinforcement Learning for Mobile Robot Position Control

Harsh Goyal
Indian Institute of Space Science and Technology
Thiruvananthapuram, Kerala 695547
ashokgoyal71071@gmail.com

Rajneesh Singh
Indian Institute of Space Science and Technology
Thiruvananthapuram, Kerala 695547
rajneesh.sc21b111@ug.iist.ac.in

Abstract

The complexity of autonomous vehicles challenges traditional control systems, making parameter tuning difficult. This has prompted exploration of alternative approaches. This paper addresses wheeled mobile robot position control using Reinforcement Learning (RL). Conducting the learning process in simulation reduces risks to actual robots and eliminates the need for explicit knowledge of the system's dynamic model. The authors have implemented and refined the RL algorithm, and analyzed the results. They design a model that interacts with the environment, using Q-learning to reach a destination place avoiding the obstacles.

1 Introduction

A robot is a machine re-programmable by a computer and capable of carrying out a complex series of actions automatically. Robotics has been very popular for the last decade. These robots play critical roles in diverse applications, ranging from industrial automation to healthcare assistance. Central to their functionality is the ability to navigate and control their position accurately within different environments. However, traditional control systems and control engineers have faced limitations in addressing the dynamic and complex nature of mobile robot navigation tasks. This led to thinking of alternative algorithms. The paper focuses on the problem of point stabilization or position control of wheeled mobile robots. Our objective is to move a robot from its current position to a given destination point avoiding various obstacles in an unknown environment using Reinforcement Learning. RL involves learning what to do and how to map situations to actions. The agent tries different things in its environment. As it get trained, it figures out the best possible actions to take from the action space to maximize the numerical reward. In our term paper, we have followed the work by Farias et al. [2] as for exploring reinforcement learning techniques for the position control of a mobile robot.

Comparable works in literature Authors have published [4] in which they have used Deep Reinforcement Learning to solve the problem for the navigation of mobile robots. This includes various aspects like obstacle avoidance, path finding, etc. And all this has been achieved using deep reinforcement learning.

In [3], the authors have used deep Q-network to make the robot learn to avoid the obstacles and to figure out an optimal path to reach its destination. They have used an RGB camera to provide the input to the robot. The agent contains three modules: image acquisition and pre processing, value function acquisition and action selection.

In [1] the authors have used ultrasonic sensors to navigate the robot. Here they have used fuzzy logic to accomplish this task.It is equipped with twenty six ultrasonic sensors, stereo cameras and structured light, and sensors for calculating the current position.

2 Model and Problem Statement

2.1 What is the environment?

We have tried to make a model for the mobile robot which has been trained to reach its destination point by tackling any obstacles that come on its way. The environment consists of all the location on which this robot can be along with the destination and set of obstacles which the robot encounters while interacting with the environment. Two models have been made to do a simulation for this problem first MDP based and second based on RL. In both of the models the environment is the same as described above. As the state space which are dealing with here is infinite therefore to simplify the tasks two different approaches have been adopted. In the models designed by us the state, action and reward spaces have been defined as

2.1.1 Action Space

We have used the set of rotational velocities of the two wheels(left and right) to describe the action space. Therefore the action space consists of four actions: right, down, left, up.

A = [1, 2, 3, 4]

Action space which has been discretised i.e. only four directions are possible. The same action space has been used in both the models.

2.1.2 Reward Space

In the first model every time the robot reaches the target it is given a reward of +100, and if the robot performs a move which does not take it to its target location, then the robot is penalized (reward of -10). So the reward space is

R = [100, -10]

In the second model robot has been given a reward of +2000 for reaching the destination, it has to pay a cost of 1000(i.e. reward = -1000) if it hits the obstacle and for all other cases it has been given a reward which is the negative of the distance of the robot from the destination.

2.1.3 State Space

In the first model we have chosen a simple 10x10 grid to store the position of the robot. This means there are 100 points in the grid where the robot can be present. Along with two different scenarios of turning clockwise and anticlockwise. Hence the state space consists of 10x10x2 = 200 points.

In the second model the state space used is of the form

[[destination distance, destination angle], [nearest obstacle distance, nearest obstacle angle]]

The destination distance and the distance to the nearest obstacle are calculated using Pythagoras theorem. And the angles are also calculated and this set is combined into a state. Here the continuous state space has been discretised such that the set of angles (0 o to 360 o) has been divided into four parts each with 90o . And the range of distance which can lie between 0 and 500 has been discretised into 500 points. Therefore the total number of states possible in this model is 4 million.

2.2 Performance Evaluation

To evaluate the performance we have compared the steps taken by the robot to reach the destination. For this performance evaluation we have compared the total steps taken by an untrained robot and our robot which has been trained using Q-learning. We think that this optimization criterion is appropriate because the main objective of this project was to design a model for the robot using reinforcement learning which can learn to reach its destination in least possible steps along with avoiding any obstacles in the way.

3 Reinforcement learning agent

3.1 Paper

In the paper we found that the task of controlling the position of the mobile robot has been done by following several different methods. In the paper the problem has been divided into two parts, in the first part agent interacts with the environment which does not contain any obstacles and in the second part an obstacle has been introduced into the environment. Both of these parts have been analysed by considering two different approaches. One of the methods used here is to control only the rotational velocity of the robot by controlling the speed of the motor which is attached to the wheels of the robot. In the second method both the rotational as well the linear velocity of the robot has been controlled.

When there is no obstacle then it has been found that the Inter Process Communication (IPC) control law performs the best and the performance of the Villela algorithm is worst whereas the RL algorithms lie in the intermediate range. Now when the obstacle has been introduced it has been noticed that the RL algorithms surpass the IPC control law and show the optimum results. For achieving this the agent has been trained for 8 million episodes.

3.2 Our Implementation

We have adopted the same model of the agent and the environment as described in the paper to solve this problem. To make the problem more interesting we have introduced more than one obstacle on the way of the robot. The training and the testing phases have been the most crucial phases in this entire mobile robot problem. Due to the extremely large size of the state space the agent has to be trained well so that it can achieve its goal. For doing this we have trained our agent for 10 million episodes and compared its performance with the comparatively less trained and also untrained models of the agents in the testing phase. It has been found that our trained agent performs remarkably well as compared to the untrained agents. A simulation for this model has been performed and the net rewards obtained after each episode has also been compared.

We have also considered a MDP model of the same problem. For this model we have reduced the length of the state space. In this implementation it has been found that the policy and the value iteration and the policy iteration converge, giving us the motivation to consider the problem in an RL setting and then evaluate the results.

Q-learning has been used in this RL implementation. The robot has been trained using an offline policy which focuses more on exploration than exploitation. After the robot has been trained using this policy, we have finally tested the robot using a behaviour policy which focuses on exploitation. This technique is also useful in real world scenarios as it avoids any damage dealt to the robot while training. Q table has been made using the pickle package in python. And the update has been made in the following manner

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \max_a Q^*(s_{t+1}, a) \tag{1}$$

$$Q^{(i+1)}(s_t, a_t) = (1-\alpha)Q^{(i)}(s_t, a_t) + \alpha \left(r(s_{t+1}, a_t) + \gamma \max_a Q^{(i)}(s_{t+1}, a) \right) \tag{2}$$

4 Extensions

There are multiple ways to solve this position control problem of mobile robots. These can also be implemented using PyTorch and TensorFlow. If considering the regime of machine learning then using neural networks is also one of the possible ways in which we can make the robot do the assigned task efficiently. One such model has also been considered where using 8 neurons.

The other options which were considered to solve this problem are:

Temporal difference : In the paper which we had chosen the position control problem has been solved using Q-learning. First a Q- matrix is created by letting the agent interact with the environment and then the final Q-matrix was exploited and it turns out that the robot performs extremely well using this method. We have also used the same algorithm as described in the paper chosen by us

TD(0) has been used along with offline policy improvement. The same problem can be extended to TD() algorithm but the problem that we encountered was the high amount of computational time which was used by these algorithms.

Function approximation : This is a very nice way to solve the mobile robot problem as it has got a very huge state space and function approximation algorithm works well with these kinds of situations. The most crucial step is the choosing of the basis functions. A better choice of basis functions can make the task of performing gradient descent very easy. So that finally gradient descent can be used for this task.

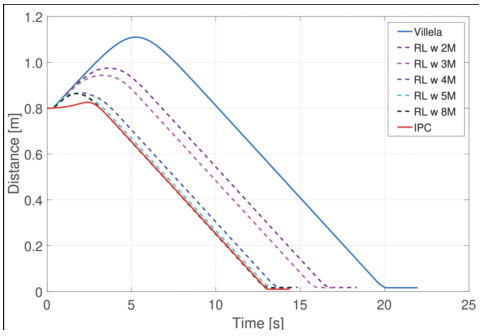
Using supervised learning is also one of the ways which we have considered in this task. For this we have used a function which supervises the agent about the action it has to take when it reaches a particular state. It has been found that when there is no obstacle in the environment then the robot performs even better than the Q-learning but when we introduce obstacles in the environment then Q-learning performs better. Implementation techniques like Deep Q-Network can also be used to solve the same problem but this requires additional work to be done in order to make the algorithm work effectively.

The papers [2] which we have used as a reference for this project also talk about certain non-RL based control algorithms which sometimes perform even better than the reinforcement learning methods. In these methods a control law is defined which performs very well for the defined states but the main issue with this kind of algorithm is their inability to learn from their experience which is something that makes reinforcement learning a better candidate for solving such problems as here the agent gets the ability to learn from the experience.

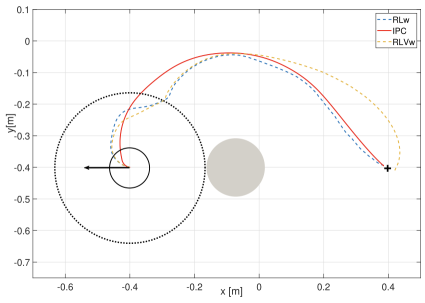
5 Performance analysis and comparison

In our project we have tried to make an agent similar to what has been described in the paper [2] . But due to some computational constraints and also the reason that we wanted to bring something new, the results which we have obtained are not exactly the same as that defined in the paper but they match well to a reasonable accuracy.

Figure 1a from the paper [2] shows that the IPC algorithm performs the best when no obstacle has been introduced into the environment. Figure 1b shows the performance of the algorithms when an obstacle has been introduced. It is noticed that in this case the RL algorithm performs even better than the IPC algorithm.



(a) Distance to the target vs. time for each control algorithm (Villela, IPC, and RLw)

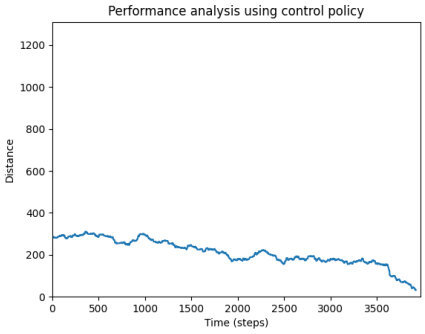


(b) Performance of the algorithms when an obstacle has been introduced

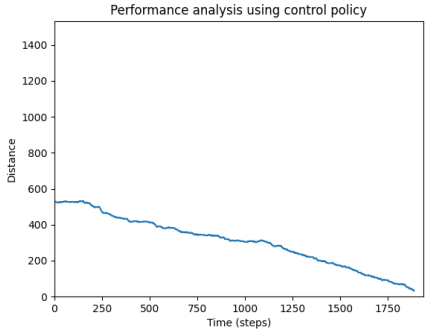
Figure 1: Comparison of algorithm performance

5.1 Performance of our agent

Figure 2a shows the performance analysis when an obstacle has been introduced into the environment and it is seen that the robot first tackles the obstacle so its distance from the destination slightly increases and then it reaches the destination in such a way that the distance in each step is minimised. Figure 2b shows the performance analysis when no obstacle has been introduced into the environment



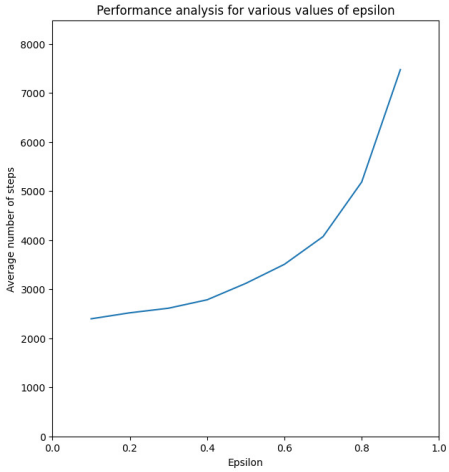
(a) Performance analysis when an obstacle has been introduced



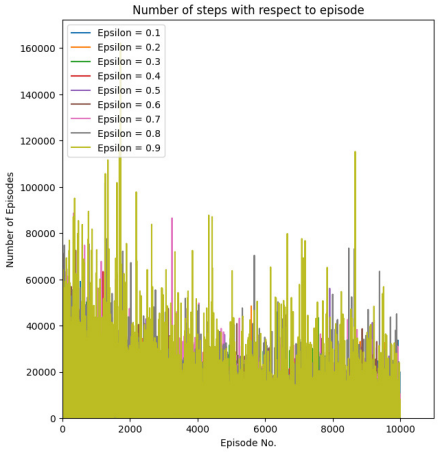
(b) Performance analysis when there is no obstacle

Figure 2: Comparison of performance analysis

and it is seen that the robot reaches the destination in such a way that the distance in each step is minimised.



(a) Performance analysis for various values of epsilon



(b) Number of steps with respect to episode for different values of epsilon

Figure 3: Comparison of epsilon-related performance analysis

Figure 3a shows as the value of epsilon increases the average number of steps also increases. This means the agent takes more steps to reach its destination point. From Figure 3b it can be noticed that as the episode no. increases the steps required by a mobile robot to reach the destination decreases and it is maximum for epsilon = 0.9 and minimum for epsilon = 0.1.

6 Challenges, ideas and future work

In our project, we faced challenges with parameter tuning, specifically the epsilon value in the epsilon-greedy algorithm. Initially, a small epsilon value with epsilon decay was implemented to balance exploration and exploitation. However, the low epsilon eventually hindered exploration by forcing the robot into exploitation mode even in unfamiliar states. Transitioning to function approximation may offer better extrapolation capabilities and address this issue in discrete state spaces.

System Model Refinement One challenge is refining the system model for simulation and control. Future work involves incorporating complex dynamics and environmental factors for a more realistic representation of the robot's operating environment.

Enhanced Reward Design Creating effective rewards is crucial for RL algorithms. We can refine rewards to encourage desired behaviors and boost learning efficiency.

Transfer Learning and Generalization Extending learned control policies to new environments or robot configurations is a significant challenge. Investigating transfer learning and generalization techniques can enhance the adaptability of the RL algorithm to diverse operating conditions and hardware setups.

Computational complexity and efficiency Training the robot is time-consuming, managed through offline training. Real-world dynamics demand continuous improvement from RL-based control systems. Scaling RL algorithms efficiently for larger spaces is challenging. Future work could focus on enabling real-time control on resource-constrained robotic platforms.

Future work may include applying RL to various control tasks for mobile robots, like navigating around bigger obstacles, following specific paths accurately, coordinating groups of robots in formations, and reaching agreements between multiple robots. RL based algorithms are also useful in space applications such as rovers to explore unknown terrain of any planet.

References

[1] Hee Rak Beom and Hyung Suck Cho. A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(3):464–477, 1995.

[2] Gonzalo Farias, Gonzalo Garcia, Guelis Montenegro, Ernesto Fabregas, Sebastián Dormido-Canto, and Sebastián Dormido. Reinforcement learning for position control problem of a mobile robot. *IEEE Access*, 8:152941–152951, 2020.

[3] Jing Xin, Huan Zhao, Ding Liu, and Minqi Li. Application of deep reinforcement learning in mobile robot path planning. In *2017 Chinese Automation Congress (CAC)*, pages 7112–7116, 2017.

[4] Kai Zhu and Tao Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021.