Path Planning Using Deep Reinforcement Learning for Mobile Robot

Sudipta Sen
Department of IT
IIIT Allahabad
Prayagraj, India
mit2021031@iiita.ac.in

***Abstract*—For a long time now autonomous driving is a great field of study and research both in academics and industry. Many big players in the industry like Tesla, Intel, AutoX, Uber are doing research in the area of autonomous driving. Path planning is important part of autonomous driving as it enables the agent or the autonomous vehicle to determine the path to reach the destination from the current location avoiding all types of obstacles. Though there is much research already done in the area but still there are many areas for improvement. It is very important to detect obstacles every time very accurately, avoid those obstacles, and automatically navigate through the environment to achieve the desired task. Not only in autonomous driving but also in any mobile agent like logistics and supply management boats, transport payload boats, surveillance boats path planning is one of the very important tasks to perform accurately**

***Index Terms*—Deep reinforcement learning, Path planning, Proximal policy optimization, mobile robots**

I. Introduction

Deep Learning is one of the most popular fields in Machine Learning research and has allowed scientists to deal with complex problems in a wide range of domains. Reinforcement Learning (RL) is built upon the principles of learning while interacting with the environment. There is no predefined dataset as supervised or unsupervised learning. In Reinforcement Learning the agent interacts with the environment and generates its data according to its policy based on the action performed the agent receives a positive or negative reward and based on the reward the agent gets to know where the performed action is good or bad and thus the agent learns. Combining deep learning and reinforcement learning the term Deep Reinforcement Learning (DRL) comes in.

The mobile robots are in the top streamer for decency. Researchers are doing serious enhancement to make robots that are as efficient as humans to perform certain tasks that may be better than humans also where there is no human inaccuracy. Mobile robots are being used in various areas as -

- Transport Payload
- Vacuum Boats
- Mobile robot for logistic-supply management
- Warehouse Management Robots
- Surveillance Boat

Deep reinforcement learning has led us to many new possibilities in solving complex navigation i.e path planning tasks. Using DRL researchers can do many improvements in obstacle avoidance and autonomous navigation task.

Areas that need to be focused on while making an autonomous mobile robot are -

- Localization:- localization is the process of determining where a mobile robot is located with respect to its environment.
- Perception:- Perception means modeling the environment. The widely used approach for perception and localization is SLAM (simultaneous localization and mapping) algorithms.
- Path Planning:- It is how autonomous vehicles plan their movements and navigate through the environment
- Motion Planning:- Motion planning is the process by which the robot defines the set of actions you need to execute to follow the path you planned.
- Trajectory Planning:- It is the same as path planning but the difference is path planning does not consider time whereas trajectory planning does.
- Avoiding Obstacles:- Obstacles can be movable or immovable.

The robot needs to represent the environment effectively for proper path planning and avoiding obstacles. Representation of the environment can be done in two ways –

- Map representation
- Configuration space representation

In map representation the size of the obstacle is similar to its original size and the robot is taken as a single point i.e robot has no size. But in configuration space representation the size of each obstacle is increased by the size of the robot and the robot is taken as a point. Map and configuration space is visualized in figure 1.1.

There are some categories of the map also for the representation of the environment –

- Recognizable Locations:- The map consists of a list of locations that can be reliably recognized by the robot. No geometric relationships can be recovered.

- Topological Map:-The map records which locations are connected by traversable paths. Connectivity between visited locations can be recovered.
- Metric Topological Maps:- Here distance and angle information is added to the path description.
- Full Metric Map:- Object locations are specified in a fixed coordinate system.

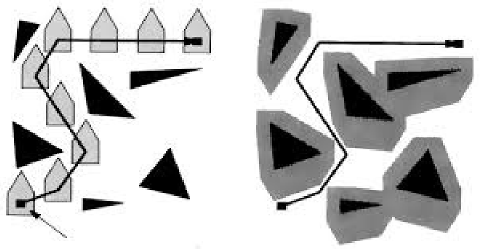

Fig. 1: Map Vs Configuration Space

In this project, our focus is on path planning only. Here we use PPO (Proximal Policy Optimization) algorithm for path planning for mobile robots and use the CV2 module of python to create an environment for the agent to interact.

II. Related Work

Path planning, has been an extremely active research area in robotics, artificial intelligence and control. Many papers have discussed methods to control the car automatically, but they require many components for full control. Creating an autonomous agent requires three main tasks—recognition, prediction and planning.

In general, the automatic exploration scheme can be divided into two categories: traditional methods and intelligent methods. The former, have been attracting considerable. interest since the 1990, employs expert features of maps for selecting the next goal point. The latter learns the control or strategy directly from the observation based on the learning approaches.

Traditional Path Planning Approaches: Path planning can be divided into two categories: global path planning and local path planning. The former approach includes graph-based approaches (for example, Dijkstra and A*) and sampling-based approaches (for example, RRT and its variant), in which all the environmental information is known to the robot before it moves. For local path planning, at least a part or almost all the information on the environment is unknown. Compared to global path planners, local navigation methods can be very effective in dynamic environments. However, since they are essentially based on the fastest descent optimization, they can easily get trapped in a local minimum. A promising solution is to combine local planning with global planning, where the local path planner is responsible for amending or optimizing the trajectory proposed by the global planner.

DRL was first applied in discrete control of mobile robots to realize obstacle avoidance. Pfeiffer et al. adopted the planning results from Dijkstra and DWA as the labels for convolutional neural networks (CNN) to train a path planner. Tai et al. proposed the end-to-end obstacle avoidance strategy of mobile robots based on deep Q learning. After that, Tai et al. presented a map-free indoor navigation system according to sparse laser radar signals and DRL. Faust et al.achieved long-range robotic navigation by combining PRM and RL. Francis et al. used PRM as the sampling-based planner, and AutoRL as the RL method in the indoor navigation context to realize long-range indoor navigation. Zeng et al. presented a jump point search improved asynchronous advantage Actor-Critic (JPS-IA3C) for robot navigation in dynamic environments, which computes an abstract path of neighboring jump points (sub-goals) by the global planner JPS+ and learns the control policies of the robots' local motion by the improved A3C (IA3C) algorithm. Here, the IA3C denotes the local planner combined with the global planner to realize a path planner for a robot. Iyer et al. proposed an approach of collision avoidance robotics via meta-learning (CARML) to explore a 2D vehicle navigation equipped with a lidar sensor and compare against a baseline TD3 solution to solve the same problem.

Inspired by the above-related works, we propose a novel incremental training mode for path planning of DRL-enabled mobile robots, to reduce the time for algorithm training and testing.

III. Path Planning Approaches

Path planning is an important primitive for autonomous mobile robots that lets robots find a path between two points. The path may or may not be optimal or shortest or has minimum cost. The various matrices to evaluate how good a path is, are done in trajectory planning. Path planning requires a map of the environment and the robot to be aware of its location concerning for to the map.

- Centralized Methods:- state space of all robots are combined into one space and then graph search algorithms are used to find the path.
- Decentralized Methods:- Robots share information about their locations and each robot plans its path taking into consideration the location of other robots.
- Shared Parameters:- Here robots share parameters of a single policy. The learning is the same but the behavior is different.

Planning concepts can be divided into two subcategories -

- Reactive Planning:- operate on a few simple rules are called reactive planners. They are also relatively short-sighted when considering what their future actions will be.
- Proactive Planning:- An entire sequence of movements or a high-level path from the robot's current position to a goal position. It assumes the robot has enough information to know exactly what it would do in the future.

Every robot uses both types of planning concepts. Proactive planning is used by robots to ensure long-term goal whereas reactive planning is used by robots to ensure short-term goal.

A path-planning (or other) algorithm is called complete if –

- Guaranteed to find a solution in a finite amount of time if a solution exists
- reports failure when a solution does not exist

IV. Methodology

A. *Problem Statement:*

A mobile robot also called an agent should reach to the destination from a starting point. In the path, there are obstacles. The agent must not collide with any of the obstacles and should reach the destination within a finite amount of time. The environment is known to the agent.

B. *Proximal Policy Optimization*

In this project path planning of a single agent system in the environment is done by the proximal policy optimization (PPO) algorithm. PPO is a policy gradient approach. Unlike popular Q-learning approaches, PPO does not learn from stored data, it learns directly from what its agent encounters in the environment. Once a batch of experience has been used to do gradient update, the experience is then discarded and the policy moves on. The policy gradient method is less sample efficient as it discards the experience as it is used once. In supervised learning, there is a fixed set of data that is used to train the model and after training the model is used in a real environment for testing. But for reinforcement learning the case is different. In DRL the agent makes interactions with its environment according to the designed policy and based on the data that is generated, it learns accordingly. The distribution of data which is based on rewards and observations is changing over time as the model learns. It causes instability in the system. DRL is also very sensitive to the initial value of the hyper-parameter that is being used in learning and testing. PPO (Proximal Policy Optimization) is outlined to overcome these types of weaknesses. The main advantages of PPO are

- Coding is easy compared to other algorithms
- Efficient sampling
- Hyperparameter tuning is easy and robust.

In PPO there are two separate neural networks are used:-

- Actor:- It selects an action on the current state based on the current policy.
- Critic:- Critic defines how good the action is and based on the critic's output the policy is further modified. Easy to tune

The loss function in the policy gradient is

$$L^{PG}(\theta) = \hat{E}_t[\log\pi_\theta(a_t|s_t)\hat{A}_t]$$

Where,
$\log\pi_\theta$ is policy network's output
$\hat{A}_t$ is the estimate of value function for the selected action
i.e $\hat{A}_t$ = discounted sum of reward - value function.
Discounted reward = $\sum_{k=1}^{\infty}\gamma^k R_{t+k}$ So the advantage function gives how our estimation is about the reward function –

- $\hat{A}_t>0$ implies our model performs better than expected.
- $\hat{A}_t<0$ implies our model performs less than expected.

This loss function suffers from many problems one of them is the update of the policy function can be too far from the old policy. It may happen that the old policy works quite well but after the new update it performs badly and it may not be possible to recover again. To overcome this the algorithm Trust Region Policy Optimization (TRPO) is developed. The optimization function in TRPO is -

$$\theta_k = \arg\max_\theta L(\theta_k, \theta)$$

subject to,
$D_{KL}(\theta \,||\, \theta_k) \leq \delta$
Where,
$L(\theta_k, \theta) = E[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_\theta}(s,a)]$
$D_{KL}(\theta \,||\, \theta_k) = E[D_{KL}(\pi_\theta(.|s) \,||\, \pi_{\theta_k}(.|s))]$

The KL constraints restrict the new policy to move far from the current policy. The problem is the KL constraint give additional overhead to our optimization process as it becomes a constraint optimization problem and can sometimes and can lead to very undesirable training behavior.

To overcome this problem PPO comes in where the constraints are embedded into the optimization problem itself. So the optimization function becomes –

$$L^{CLIP}(\theta) = \hat{E}_t\,[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

Where,
$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$

Here $r_t(\theta)$¿1 means the action is more likely to be taken in the new policy than the previous one and $r_t(\theta)$¡1 if the action is less likely in the current policy.
The expectation operator in the objective function means it going to calculate over batches of trajectories. The first term pushes the policy toward action that yields a high positive advantage function over the baseline. The second operator is just a clipping function for $r_t(\theta)$ between 1-ϵ to 1+ϵ.

This loss function restricts the update from -ϵ to +ϵ. But the problem is if the advantage function is negative then always it will take the -ϵ so the policy will decrease slowly. To restrict this the final objective function for model training is -

$$L^{CLIP+VF+S}(\theta) = \hat{E}_t\,[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

In this final loss function:

- The second term is updating the baseline network. The baseline network estimate returns the average estimate of discounted reward from this specific state. Since value and policy are part of the same computation graph so we can combine it into a single loss function
- The last term in the loss function is the entropy term. This term is ensuring that the agent does enough exploration during training.

C. *MDP for path planning using PPO:*

The four components for defining the MDP are –

- The states are given by the sensors - [0,1,2,3,4,values]
- The action is to steering the wheel from -4 to +4 degree
- The reward function for path planning is formulated as –
  2.0 + (.8 if s[0]¡2 else 0.01) + (.6 if s[1]¡2 else 0.01) if no collision
  -1 for collision

The reward function is the most important thing in the DRL training as based on the reward only the algorithm knows how good the action is and it tries to maximize the overall reward and tries to minimize the negative rewards earned.

D. Actor design

- Dense layer with 200 neurons with activation function Relu and batch normalization
- Dense layer with 100 neurons with activation function Relu and batch normalization
- In output there are two neuron
  one neuron with tanh activation function
  another neuron with softmax activation function

E. Critic design

- Dense layer with a number of neurons 200 and activation function Relu and batch normalization
- Dense layer with a number of neurons 100 and activation function Relu and batch normalization
- In output there is one neuron with no activation function

F. Other hyper parameters

- Discount factor = 0.9
- Actor learning rate = 0.0002
- Critic learning rate = 0.0002
- Batch size of each iteration = 64
- Number of training iterations with each mini-batch for the actor = 8
- Number of training iterations with each mini-batch for the critic = 8
- Adam Optimizer is used for the optimization

V. Result

The environment that is used to train the model is given in Figure 5.1
After training for 1000 epochs the reward and episodes graph is given in Figure 5.2

As we can see in figure 5.2 as the number of epochs in training are increasing the reward is also increasing means the agent is learning to increase the reward and avoid colliding with obstacles and windows and also the agent is reaching the destination also. Here the maximum number of steps that can be taken to reach the destination are 800 and if the agent reaches the maximum steps the model will receive a negative reward. So the model is learning to reach the destination and maximizes the reward also.

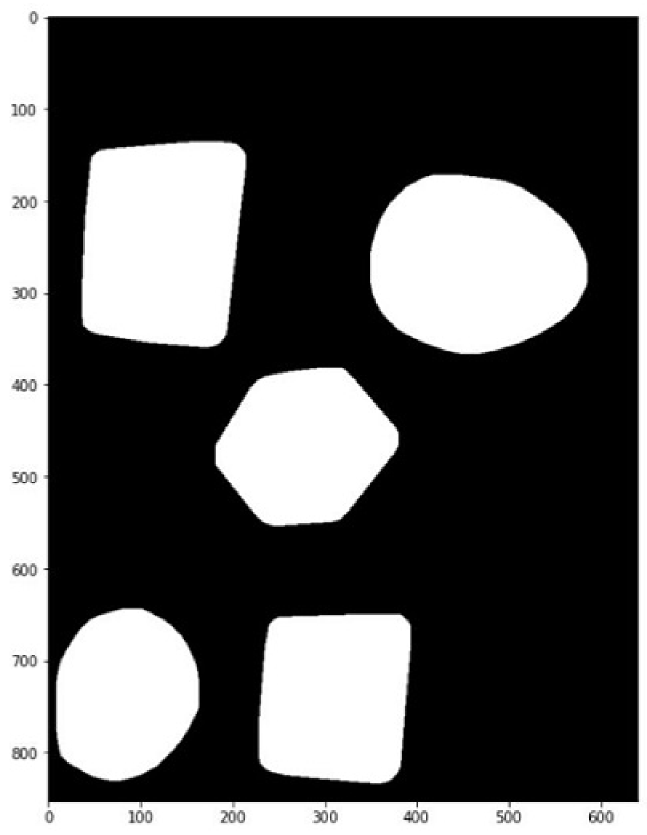

Fig. 2: Environment

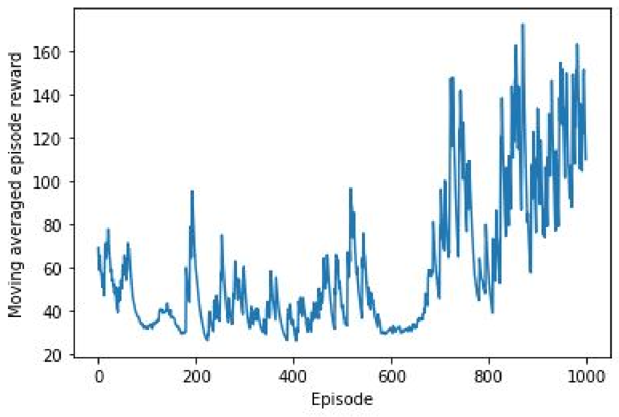

Fig. 3: Reward and Episodes graph

VI. Conclusion and Future Work

From the result we can see that the PPO algorithm is working very well for path planning for a single mobile agents. There are Q-learning algorithms also for path planning but they are offline algorithms, PPO is an online algorithm means PPO trains the model on a batch of experience and after that is discards the old data and again train the model on a new batch o experience. Here we discussed all the model architectures to build in our model with PPO. We have evaluated this model a huge number of times and the model performance is satisfactory. If we increase the training time the model will work more accurately. In this project the obstacles are static so there is also a scope for trying this algorithm with moving obstacles. We can extend the algorithm to support multi-agent also.

REFERENCES

[1] Michael W. Otte "A Survey of Machine Learning Approaches to Robotic Path-Planning"

[2] : B Ravi Kiran, Ibrahim Sobh2, Victor Talpaert3, Patrick Mannion4, Ahmad A. Al Sallab2, Senthil Yogamani, Patrick Pérez "Deep Reinforcement Learning for Autonomous Driving: A Survey"

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov "Proximal Policy Optimization Algorithms"

[4] Nut Chukamphaeng, Kitsuchart Pasupa, Martin Antenreiter, Peter Auer "Learning to Drive with Deep Reinforcement Learning"

[5] Anh T. Huynh, Ba-Tung Nguyen, Hoai-Thu Nguyen, Sang Vu and Hien D. Nguyen "A Method of Deep Reinforcement Learning for Simulation of Autonomous Vehicle Control"

[6] Martin Gromniak, Jonas Stenzel "Deep Reinforcement Learning for Mobile Robot Navigation"

[7] Junli Gao, Weijie Ye, Jing Guo and Zhongjuan Li "Deep Reinforcement Learning for Indoor Mobile Robot Path Planning"

[8] Abdur R. Fayjie, Sabir Hossain, Doukhi Oualid, Deok-Jin Lee "Driverless Car: Autonomous Driving Using Deep Reinforcement Learning In Urban Environment "