Mapless Navigation Using Deep-RL Algorithms

Lyuheng Yuan
lyuheng@seas.upenn.edu

***Abstract*—We want to represent a path planner based on Deep Deterministic Policy Gradient Deep (DDPG), a Reinforcement Learning (RL) algorithm, with respect to the mobile coordinate frame as input and continuous steering as output. Traditional motion planners mostly depend on the prior obstacle map of the navigation environment where laser sensors should be highly precise, costing a lot to build the obstacle map of the environment as well. We show that through Deep-RL methods, we need not any manually designed features such as obstacle maps. The trained end-to-end planner can be applied in unseen virtual environment without any collision with any obstacles. Code and demos are available at https://github.com/lyuheng/650project.**

***Index Terms*—motion planner, Reinforcement Learning, DDPG**

I. Introduction and Motivation

Nowadays, low-cost methods, like WiFi localization [1] and visible-light communication [2], provide lightweight solutions for mobile robot localization. Thus, mobile robots have easy access to the real-time target position with respect to the robot coordinate frame. And it is really challenging for a motion planner to generate global navigation behaviors with the local observation and the target position information directly without a global obstacle map.

Thus, we are motivated to present a learning-based mapless motion planner in this project. This motion planner takes only 10-dimensional sparse range findings and target relative information as references. In Gazebo virtual environments, a nonholonomic differential drive robot was trained to learn how to arrive at the target position with obstacle avoidance through deep reinforcement learning [3]. Three essential parts in our project are Deep RL in robotics, mapless navigation and Gazebo environments.

A. Deep Reinforcement Learning in robots

The applications of deep reinforcement learning in robotics are mostly limited in manipulation where the workspace is fully observable and stable. In terms of mobile robots, the complicated environments enlarge the sample space extremely while deep-RL methods normally sample the action from a discrete space to simplify the problem which is not accurate in some cases. We focus on the navigation problem with continuous control of deep-RL, which is the essential ability for the most widely used robot [4].

B. Sample based & Mapless navigation

As discussed in class, RRT* are sample based motion planning algorithm that replies on prior obstacle map of the environment. There are two issues for this task: (1) the time-consuming building and updating of the obstacle map, and (2) the high dependence on the precise dense laser sensor for the mapping work and the local map prediction. Learning based navigation can alleviate the dependence of prior obstacle map and shows generality when transferring into moving obstacles environment.

C. Gazebo environments

Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. Gazebo provides a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces.

With Gazebo we are able to create a 3D scenario on computer with robots, obstacles and many other objects. Gazebo also uses a physical engine for illumination, gravity, inertia, etc. In the meantime, ROS serves as the interface for the robot. Combining both results in a powerful robot simulator.

II. Related Work

A. Obstacle-Avoidance-based Navigation

LeCun *et al.* [5] described a vision-based obstacle avoidance system for off-road mobile robots. The system is trained from end to end to map raw input images to steering angles. It is trained in supervised mode to predict the steering angles provided by a human driver. A remote computer processes the video and controls the robot via radio. The learning system is a large 6-layer convolutional network. They applied this end-to-end learning methods to the task of obstacle avoidance for off-road robots. The architecture of the system allowed it to learn low-level and high-level features that reliably predicted the bearing of traversible areas in the visual field.

B. Deep Reinforcement Learning

While reinforcement learning agents have achieved some successes [9], their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. They use recent advances in training deep neural networks [10] to develop a novel artificial agent, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning.
Reinforcement learning has been widely applied in robotic tasks [6]. Mnih *et al.* [7] utilized deep neural networks for the function estimation of value-based reinforcement learning

which was called deep Q-network (DQN). DQN works well in discrete action space. When comes to continuous action space, Lillicrap *et al.* [12] proposed Deep Deterministic Policy Gradient (DDPG) to use deep neural network on Actor Critic method. Fujimoto *et al.* [13] proposed TD3 which addresses the value function bias by using twin critics. Our focus is to use low-dimensional sensor findings to do navigation on continuous real world problems.

III. Method and Implementation

A. Problem setting

We aim to find a mapless motion planner on a 4×4 ground with unseen obstacles, i.e. we try to simulate a function:

$$v_t = f(x_t, p_t, v_{t-1})$$

where x_t is 10 dimensional sensor findings, p_t is the relative target where we represent it in polar coorindate. v_{t-1} is the velocity at last timestep. Here v_t represents linear and angular velocity. We set max linear velocity to 0.5 m/s and max angular velocity to 0.5 rad/s to ensure safe moves. They can be regarded as inner state s_t of robot.

B. Network architecture

In this RL problem, we use DDPG as mentioned before to train our model. We use the same structure as Tai *et al.* [4].

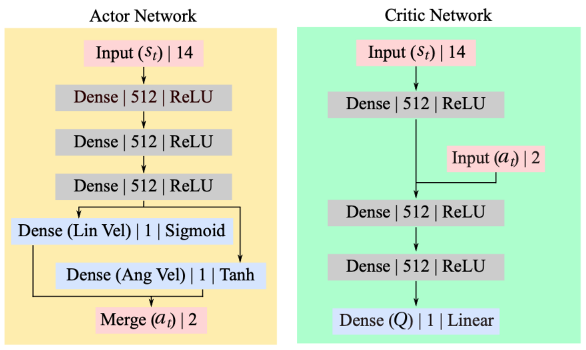


Fig. 1: The architecture of DDPG model. Net type, dimension and activation function are shown. Dense means fully-connected network.

For actor network, we normalize findings to $(0, 1)$. To constrain the velocity, we use *tanh* to angular velocity output and *sigmoid* function to linear velocity output. For critic network, we utilize 3 fully-connected network to compute Q-value.

C. Reward Function

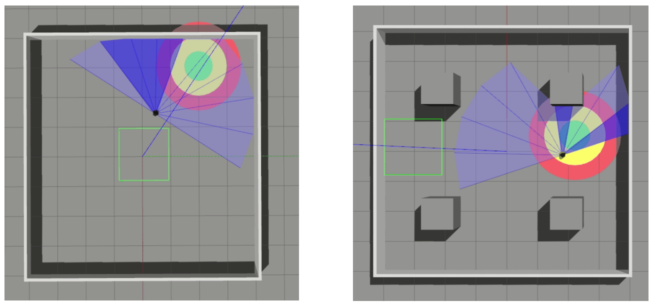
There are three different conditions in our problem setting:

$$r(s_t, a_t) = \begin{cases} r_{arrive} = 120 & if \quad d_t < c_d \\ r_{collision} = -100 & if \quad min_{x_t} < c_o \\ 500(d_{t-1} - d_t) \end{cases}$$

When robot arrives the target within a threshold, we give it a positive reward, but if robot collides with any obstacle, a negative reward is arranged. Otherwise, we compare the distance from target with the last timestamp, compute difference and multiply a hyper-parameter which motivates robot getting closer towards target. Here we choose 500.

IV. Results

We train our agent in Gazebo environment using Deep Deterministic Policy Gradient (DDPG) to control turtlebot to reach the target (green circle) without any collision in a continuous action space.
Firstly, we pretrain the model in a space without any obstacles (See Fig. 2(a)). By doing this, agent learns how to reach target in fewest steps in order to maximize reward so that it can have a 'good performance' to begin with. Training curve is shown as follows.



(a) No Obstacle Env. (b) 4 Cubic Obstacles Env.

Fig. 2: **Left**:In Env.(a), we pretrain the agent using DDPG in environment with no obstacles inside, just have 4 wall as the boundary. **Right**:In Env.(b), four square obstacles are set on four corners

Then we use a more complicated environment to fine-tunning our agent by uniformly place four cubes at four corners (See Fig. 2(b)). So the agent can learn how to avoid obstacles and use fewest steps reaching target in the meantime. Training curve is shown as follows.

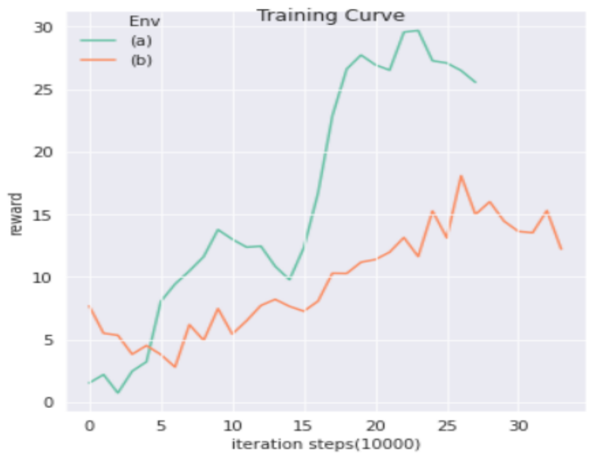
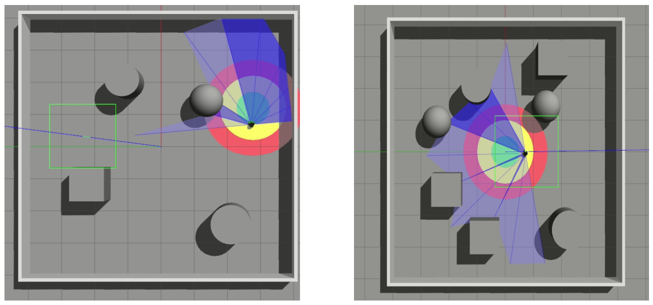


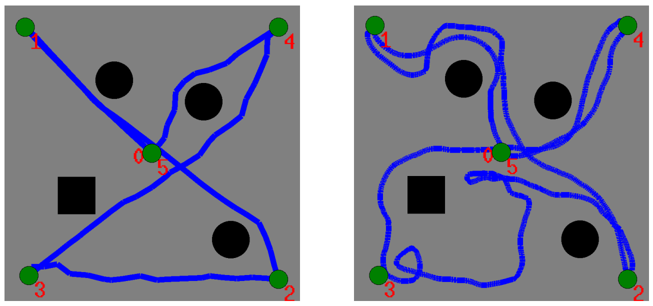
Fig. 3: Training curve in Env.(a) and Env.(b), Env.(b) is fine-tuned on Env.(a)

After the robot can reach the target quickly without colliding with the obstacles, we make the environment more complicated by replacing part of cubic obstacles with different shaped obstacles, for example, sphere, cylinder. We generate obstacles at random positions (See Fig. 4(a)). By doing this, we can see how agent perform in such unseen, complex environment. Result shows that agent can reach target with 100% accuracy.



(a) Unseen Env.1 (b) Unseen Env.2

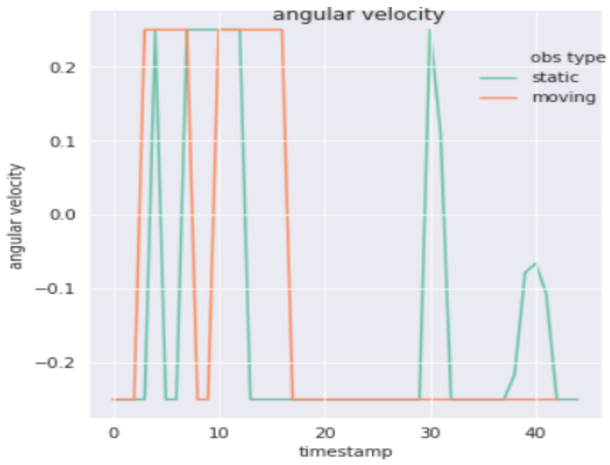
Fig. 4: **Left**: In Env.(a), we evaluate agent in a virtual unseen environment with different shaped obstacles. **Right**: In Env.(b), We evaluate in a more involved environment with more obstacles and narrower path space.



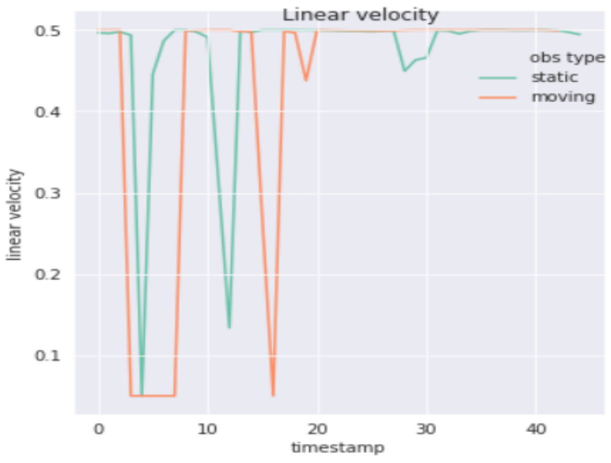
(a) Baseline: RRT* (b) DDPG Agent

Fig. 5: Numbers in figure represent the arriving order of destinations of robot. Trajectories of our DDPG Agent and RRT* are generated by evaluating on Unseen Env.1(See Fig. 4(a)). **Left**: In (a), we use RRT* as a baseline if agent knows the prior obstacle map beforehand. **Right**: In (b), agent reaches every target with 100% accuracy.

Robot tends to do rotation or swing back and forth when it hasn't decided a path straightly towards target, so we can see some circles or U-turns from the trajectories. That can be alleviated by increasing training time or fine-tunning on a different context. In addition, some trajectories in narrow space is tortuous comparing to baseline and that may be relevant to the characteristics of DDPG. DDPG is hard to converge because of its deterministic.
Since we robot performs relatively well in Unseen Env.1 (See Fig. 4(a)), we evaluate agent with an even more involved environment (See Fig. 4(b)) which has more obstacles in shape of cube, sphere and cylinder where path is narrower than Unseen Env.1. It shows that the robot can reach the target eventually although it will take longer time than the simulation before in Unseen Env.1. Demos are shown in Github.



(a) Angular Velocity



(b) Linear Velocity

Fig. 6: The figure shows the reaction of agent towards static cubic obstacle versus moving cubic obstacle ahead. We can see that angular velocity in moving condition changes ahead of static condition which means agent detects the moving of obstacles from sensor findings and divert itself in advance. Similarly for linear velocity, agent slows down for longer time meaning it hasn't decided a possible path because when agent tries to find a path, its linear velocity reduced.

Since agent can update its velocity in high frequency, we infer that it can avoid collision with moving obstacles that has relative low speed (0.2 m/s in our case). Results show that agent can detect unusual conditions and make decisions in advance compared to static obstacles as in Fig. 6. Demos are shown in Github.

V. Future Work

Simulations in the virtual Gazebo environment proved that the deep-RL trained mapless motion planner can be transferred directly to unseen environments. In this project, we trained and evaluated our DDPG trained model in five different situations and the robot can reach the target in a new environment successfully with 100 % accuracy after fine-tunning.

However, we are not aiming to replace the map-based motion planner: it is obvious when the application scenario is a large-scale and complex environment, the map of the environment can always provide a reliable navigation path. Our goal is to provide a low-cost solution for an indoor service robot with several range sensors, like a light-weight sweeping robot.

In the simulation process, we find that the robot has a poor performance in the narrow environments, sometimes the robot was not able to go across the narrow route smoothly. A possible explanation is that the network has neither the memory of the previous observation nor the long-term prediction ability. Thus LSTM and RNN [11] are possible solutions for that problem. We set this revision as future work.

References

[1] Y. Sun, M. Liu, and M.-H. MENG, "Wifi signal strength-based robot indoor localization,"in Information and Automation(ICIA),2014 IEEE International Conference on. IEEE, 2014, pp. 250–256.

[2] K. Qiu, F. Zhang, and M. Liu, "Let the light guide us: Vlc-based localization," IEEE Robotics Automation Magazine, vol. 23, no. 4, pp. 174–183, 2016.

[3] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in International Conference on Machine Learning, 2016.

[4] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," CoRR, vol. abs/1703.00420, 2017.

[5] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in NIPS, 2005, pp. 739–746.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238–1274, 2013.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[8] Sutton, R. Barto, A. Reinforcement Learning: An Introduction (MIT Press, 1998).

[9] Riedmiller, M., Gabel, T., Hafner, R. Lange, S. Reinforcement learning for robot soccer. Auton. Robots 27, 55–73 (2009).

[10] Bengio, Y. Learning deep architectures for AI. Foundations and Trends in Machine Learning 2, 1–127 (2009).

[11] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," arXiv preprint arXiv:1507.06527, 2015.

[12] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.

[13] Fujimoto S, Van Hoof H, Meger D. Addressing function approximation error in actor-critic methods[J]. arXiv preprint arXiv:1802.09477, 2018.