Finding the Shortest Path Using Reinforcement Learning

LINGFENG ZHANG, WENHAN HAO, & TIANMING QIU

Group: applied-rl17 / E3

1 Introduction

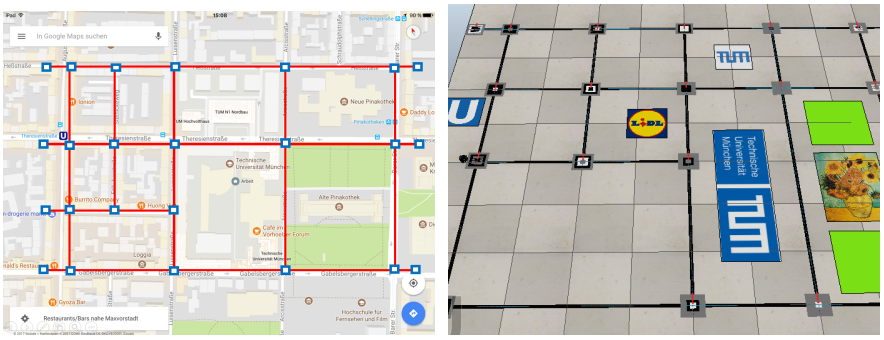
Reinforcement learning is a powerful framwork for robot to learn novel tasks by self-practice. Through trial-and-error learning, robotic agent has the chance to be exploited in a wide range of situations, including industrial and social applications. In this project, we attempt to employ reinforcement learning for path planning.

Path planning is a basic and widely-studied problem. Also it has many actual scenarios, for example, autonomous car navigation and drones for logistics. Typically, a optimal path between two locations on traffic map is computed by considering both distance and traffic situation. But if prior informantion for road networks is absent, the planning is essentially impossible. Therefore, we attempt to empoly reinforcement learning to plan path from scratch.

In order to make our project more realistic, we use the map around TUM as the training map. The original map (left) and the map in simulator (right) are shown in Figure 1. The simulation map contains several guide lines on the ground as the roads to form the road network with many crossroads. Two points on simulation map could be chosen as starting point and destination, and then agent attempt to discover the optimal path between those two points. In our project, the e-puck mobile robot takes on a role of explorer.

2 Modeling

To make our tasks tractable, we consider a grid-based representation with discrete states and action. The navigational task for our mobile robots could



(a) map from google map (b) map in v-rep

Figure 1: Training maps

be projected into this presentation by employing a number of actions such as 'drive east', 'drive west', 'drive north' as well as 'drive south' that only use a low-level controller that takes care of accelerating, moving, turning and stopping. The different crossroad in the simulated map will be considered as states. The simulation map we used has 22 states. In our experiment, we assign a reward '+10' at the destination state and '-1' at the other states. Also at the blind alley it will get a very poor reward such as '-10' in order to let agent avoid those states. At the begin of each simulation, two points will set up as the start point and destination, and the e-puck robot will be placed in the start point of the training map, and start to explore the training map. From one state to adjacent state, the robot will drive along the line we drawn on the ground. The under sensors on e-puck will scan the ground all the time in order to keep robot following the line and detect the label we use to distinguish state. Once e-puck reaches the terminal state(destination), it will be replaced in the same start point and start a new exploration.

3 Algorithm

The reinforcement learning approaches can be categorized into two groups: model-free method, which directly learns a control policy from system interaction, and model-based method, which first learns a model of the system dynamics, and then optimizes the policy under this model. At the beginning of our project, we attempt to set up a model of our environment in order to make learning process more efficient. We let all transition probability be deterministic, i.e, the results of agent performs an action are deterministic. The results are only depends on current state and performed action. And then we optimized policy under this policy. But this model is too coarse and ideal. It doesn't have ability to handle with model uncertainty and complex

situation. As small model errors accumulate, the simulated results can be big different from real situation. For this reason, we modified our plan, a typical Q-learning algorithm is used. The agent first observes current state, and then takes action accorading to policy. After the action is performed, the agent observes a reward and new state. Then the Q-value is updated by maximizing Q-value of next step.

4 Milestones

In the end we have achieved five main Milestones: determination of the mathematical model, familiar with the software environment, viable physical environment design, successful program implementation on Vrep, and stably interaction with real world. And there is also a remaining feature: state detection with front camera.

Determination of the mathematical model

Before we chose the road crossing as states, and we assumed the actions as 'turn right', 'turn left' and so on. However, it caused a serious problem that the model could not satisfy Markov Property. Assume Epuck comes to a road crossing several times, first time it comes from north, namely the last state is in the north direction of current state, and second time comes from south. Since we treat this road crossing as only one state, if Epuck both choose the action 'turn right', the transition probability of the next state will be totally different and somehow depends on not only the current state, but also the state before, like $P(s_{t+1}|s_t, s_{t-1}, a_t)$. Hence we choose a new set of actions as 'go east', 'go west', 'go north' and 'go south'. And on the high level of this problem we just treat our Epuck as a mass point. And leave the detailed problem of action realizations to the lower level. This approach make sure the scene satisfies Markov property.

Learning Python and Vrep

We don't have any experience on python before, so we have to learn it from scratch. The interface of vrep is also a hard problem for us. We spent a lot of time to manipulate it skillfully.

A viable simulation map and state detection design

One challenge we faced was that it is very difficult to detect different states in consideration of Epuck's poor performances on sensors. Finally we invent a kind of 'ternary encoder unit', which is shown in figure 2. This state detection unit will be put at each crossroads to let Epuck detect and realize where it is. So the outside stripe is used to make Epuck to stop

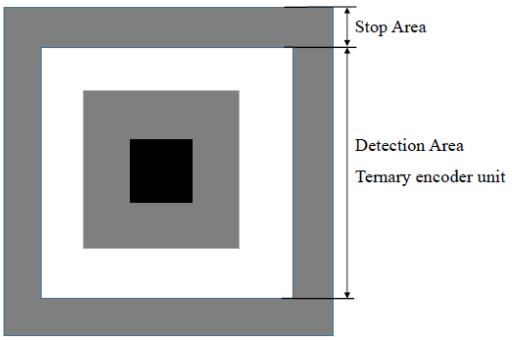


Figure 2: An example of state with ternary encoder unit

tracking the line and start state detection procedure. And the inside part contains three colors stripe and formed as a big square which could let Epuck smoothly detect while it comes from any directions to this crossroad. Afterwards Epuck will step forward for a corresponding fixed distance and read the color three times. Black is represented as '0', gray is '1' and white is '2'. With reading different colors it can get three digits in the form of ternary code. After turning 3 digits ternary code into decimal code, epuck will know which state it is. The three floor sensors under Epuck are also applied to detect the road, and the road crossings are represented as ternary encoder unit in order to distinguish between different states. Although the floor sensor are not very accurate to distinguish high resolution color space, it could be sensitive enough to detect black, white, and gray these three situations. The length of our encoder is 3. So, it can perform $3^3 = 27$ different states.

Simulation implemention on Vrep

The code can be basically divided into three main parts: Q-learning algorithm, Motion or action control and perception part. Q-learning part is designed to select the next action and update Q-value functions. Motion control is used for the low level control issues such as 'go east', turn a corner and so on. Perception focus on the three floor sensors to make sure Epuck could follow the line, stop correctly at the crossing roads, and determine current state.

Training and implementation in real physical environment

In the physical environment implementation stage we choose bluetooth communication. Although the data transmission is limited by the bandwidth, we still can achieve stable and efficient remote control through bluetooth since we don't need to transfer images. Considering the time delay

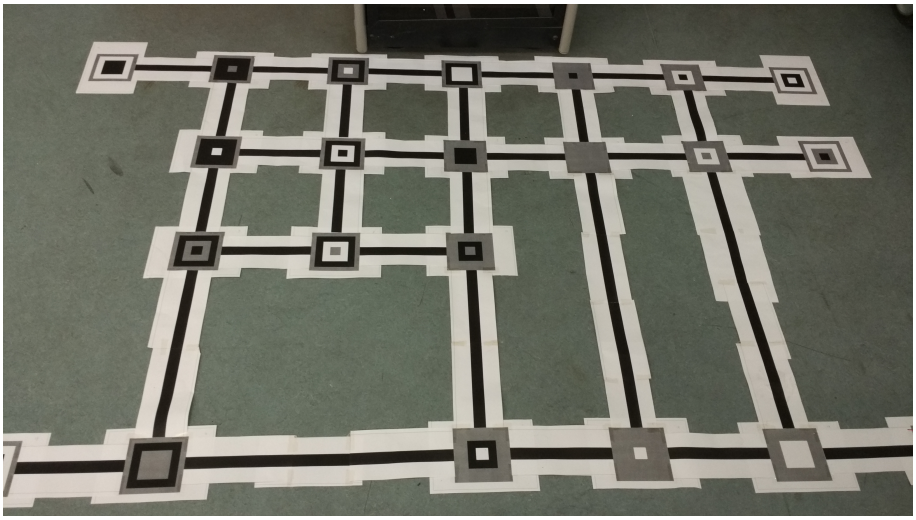


Figure 3: Implementation on a real map

we have also carried out many tests to make sure that we can drive the Epuck smoothly based on a fixed bandwidth. Another advantage is we do not need to make a lot of changes with the code, especially the APIs. The procedure is also very similar to the remote control on Vrep so we could adjust the parameter or debug more easily.

We printed the states and set up the real map on the ground which is shown in figure 3. At first we test several patterns of states to determine which gray color is the best one and how wide each stripe could be. Epuck can distinguish road and different states with the help of three floor sensors. Three floor sensors scan the ground all thee time in order to detect the change of colors on the ground.

In the real physical map there are some new problems. For example, Epuck may be blocked at the joint of roads, since the actuator performs not as acurate and strong as the one in simulation and the road is not extremely flat due to the coarse map made by paper. Sometimes communication lost may occur that Epuck could not get the commands timely and rushed out of the map. Training process takes a long time not only because of the dimensions of states but also because each successful episode of training is not easy to finish due to the above problems.

Based on the mentioned reality, we decide to combine the simulation and real world training together to achieve the convergence of Q table more quickly and efficient. And finaly the Epuck can implement an optimal path on the real map as we supposed.

Remaining task: state detection with front camera

State detection is a key component of our project. At the beginning we attempted to achieve this feature using front camera. Due to poor resolution and slow transmission we discarded this plan, and quickly changed to use floor sensor instead.

5 Conclusion

After successfully completing this course, we master some basics of reinforcement learning methods and learn how to model real engineering problems using reinforcement learning methods. Using this knowledge we constructed and implemented reinforcement learning algorithms to solve simple robotics problems on the e-Puck platform. And through practice we are also familiar with python and vrep.