Navigation of Mobile Robot

Jing Liang

Abstraction:

The project built a collision avoidance system using Q-learning for mobile robot especially Turtlebot. Open AI and Gym packages are used to provide interfaces among ROS (Robot Operating System), Gazebo simulator and Q-learning python scripts. The result of the project is making turtlebot navigate itself forward and at mean time avoid obstacles.

Introduction:

The collision avoidance is always a fundamental and big problem in motion planning area. There are many methods for the problem. With the computational ability of computers improving, reinforcement learning is becoming more and more popular in recent years, especially in auto driving area.

Reinforcement learning is one of control method. It is well used in sophisticated situation and hard-to-engineer behaviors. In reinforcement learning, agent is learning by interacting with environment to collect most reward. The categories of reinforcement learning includes Value based RL, policy based RL, and Actor Critic RL. In this project a model-free reinforcement algorihtm, Q-learning, is used.

Open AI gym is a useful tookit for reinforcemnet learning research. It provides a convenient interface among ROS, Gazebo and algorithms. The package separate the reinforcement learning system into several blocks inlcuding environment, robot and also reinforcement learning algorithm. The open ai gym package is used in the project, it cooperates with ROS to navigate turtlebot in Gazebo simulator.

code link: https://github.com/jingGM/ENPM808F.git

Architecture:

As figure 1 shows, the project consists of three main software blocks: openai, ROS and Gazebo. Openai developed an environment, which interact with ROS system. The ROS system also communicate with Gazebo to get the simulation information.

OpenAI developed an environment and provides interface with algorithm. The openai package also in charge of communication with ROS system, which could generate simulation data to certain format used by algorihtm and also provide instructions which ROS system could use.

The ROS system is platform for communication of different terminals. In the project, it communicates with Gazebo simulator to track simulation result and also provide instructions to simulator, it also communicate with openai package to get reinforcement learning algorithm done.

The Gazebo simulator create a stable simulating environment for the project. In the project, there is a world file setting a maze, and there is also a turtlebot which has a hokuyo lidar on top. The simulator would simulate the movement of turtlebot and also provide relative environment information to ROS system.

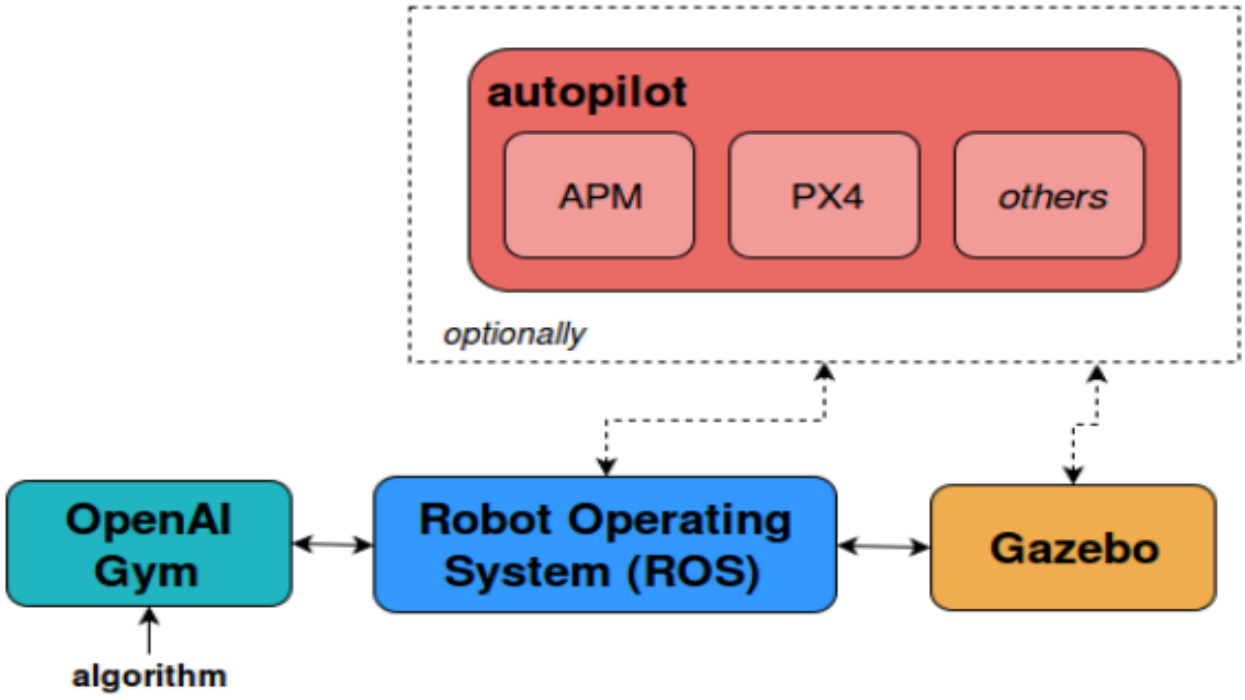


Figure 1 Architecture of the system

code link: https://github.com/jingGM/ENPM808F.git

Q-Learning:

Q-learning in the project is a value-based reinforcement learning algorithm, and since it is also model-free algorithm, it could be used for different robots. Q-learning is used to generate optimal actions regarding to different state inputs.

Q-learning will update the value in each episode using the following function:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max\alpha, Q(s_{t+1}, a_t) - Q(s_t, a_t)]$$

Q(st,at) is the Q value of current state which composed of current observation and action taken. Alpha is learning rate which decides in what rate the system is learning. R is the reward which generated by environment according to different policy. Q(st+1,at) is the value of next state, which makes the Q-learning consider future effect of each action.

There are several methods to map states to Q values. For complex environment or large number of states, neural networks is recommended, and for small number of state Q-table also works.

In the project, Q-table is used since the environment is simple and number of states is small. Here the state is only lidar values combined with actions.

In the project, the epsilon is set to 0.5, which means the robot would have 50 percent possibility to choose a random move in training, which makes equal weight of both exploration and exploitation.

The gamma value is set to 0.7 which is discount factor to decide how much weight to put to prediction of future rewards.

Learning rate is set to 0.5, which means how fast the q-learning is learning and how much in detail the q-learning is doing. The smaller value will cost long time learning but good converge, but a big value will also cause a unconverge with fast learning. Here 0.5 value is fairly good regarding training time.

The reward for each action is set as 5 scores when going forward and 1 score when do turning and -200 scores when the robot is running into collision which is decided by distance value from lidar.

code link: https://github.com/jingGM/ENPM808F.git

Simulation:

Simulation is based on three parts: Gazebo simulator, ROS platform and OpenAI package. Gazebo create simulation to real robot and certain environment. ROS provides topics to transfer data between algorithm and simulator.

Figure2 is the environment generated by Gazebo simulator. The environment is generated deliberately as a maze.

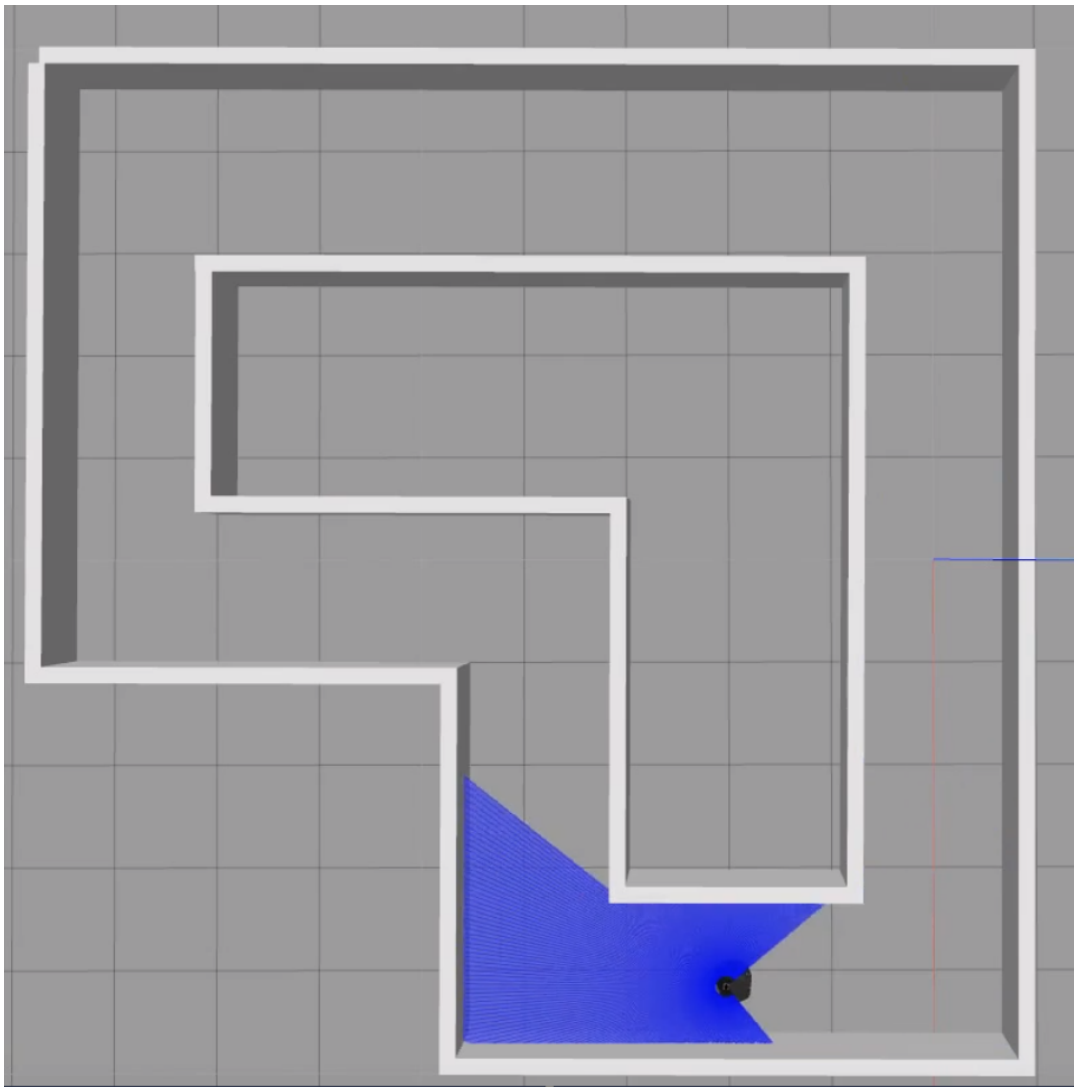


Figure 2. Simulation Environment

code link: https://github.com/jingGM/ENPM808F.git

The robot used in the project is turtlebot2 as figure 3. In the configuration of the robot, it runs at 0.3m/s when moving forward; at 0.1m/s as linear velocity when turning with 0.3r/s as angular velocity. The robot can only move forward, turning left or turning right.

On top of the robot, there is a Hokuyo Lidar used to detect the distance from robot to obstacles nearby. In simulation, the range of the lidar is around 240 degree and it's evenly separated into 10 sectors. In each sector, the value of distance would be remapped to an integer value. The sectors are: 0m-0.3m, 0.3m-1m, 1m-1.5m, 1.5m-infinit. For the ranges, distance would be mapped to values [3, 2, 1, 0]. when the lidar gives value 3, it means the robot is running into collision and the system is going to be reset and algorithm is going to do another episode of training.

In each sector of Lidar detection, the value is chosen by minimum value in the sector.

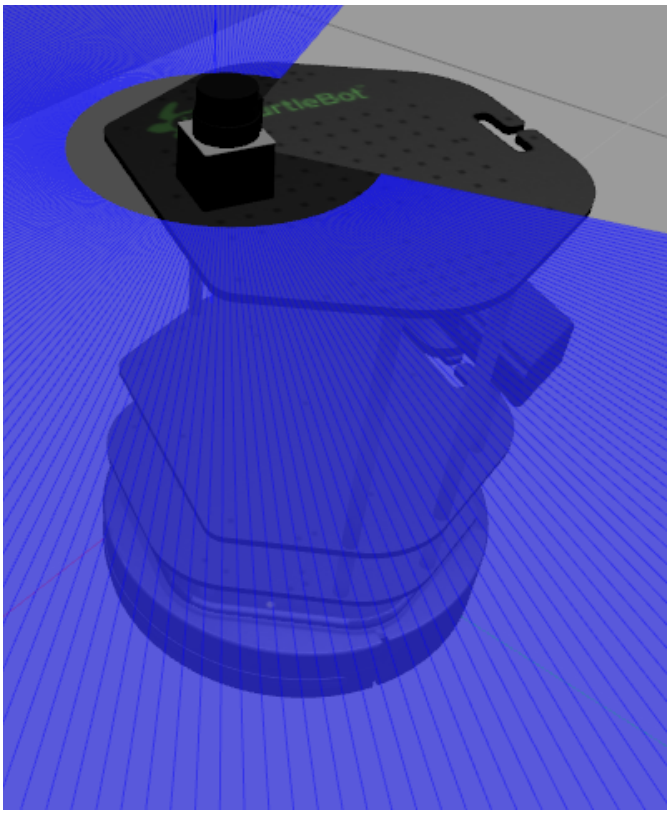


Figure 3. Turtlebot 2 with Lidar

In the ROS platform, there are two major topics created: lidar data and comtrol signal of robot. The lidar data indicates the environments and control signal provide instruction to robot from algorithm.

code link: https://github.com/jingGM/ENPM808F.git

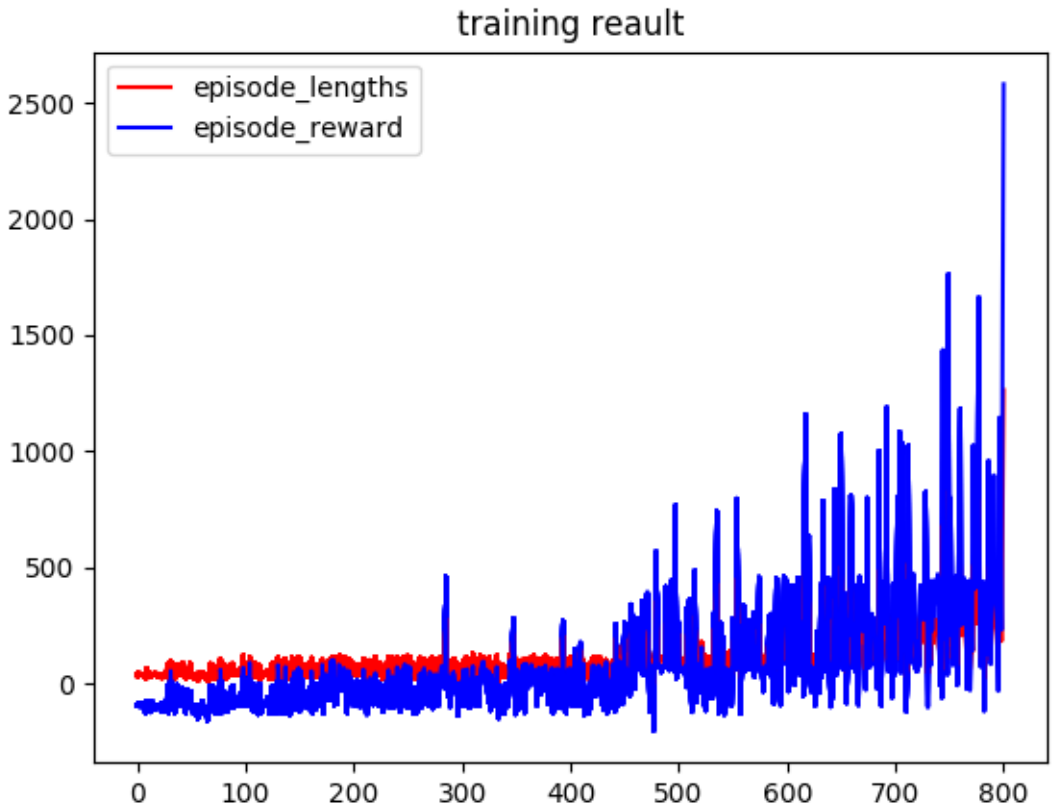
The openAI package provide data structure to store data of environment and also robot, meanwhile the package also translate data to both Q-learning algorithm and Gazebo simulator.

Result and Analysis:

After 1000 times iteration, the performance is shown as following figure 4.

The blue lines and red lines are reward and lengths robot moved in each episode, from the figure, we can see that with the iteration increasing, the reward is also increasing and length is also increasing. When the reward goes up to 2500, there are several loops the robot has run in the maze, which means it mastered the ability to travel without colliding.

Using the testing file with Q values, we can also see that the robot has the ability to run and avoid obstacles at the same time.



code link: https://github.com/jingGM/ENPM808F.git

Figure 4. Training result

Conclusion:

In the project, Q-learning is used to training a robot moving in a maze and at mean time to avoid obstacles. After the training of 1000 iterations, the robot could successfully finish a circle to starting point and also move further.

Future Work:

The future work is going to use camera to capture more information about the environment, and use neural network to train it, since now the robot has much oscillation only with lidar.

There could also be other on-policy reinforcement learning algorithms like PPO.

Reference:

[1] Zamora, I., Lopez, N.G., Vilches, V.M. and Cordero, A.H., 2016. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. arXiv preprint arXiv:1608.05742.

code link: https://github.com/jingGM/ENPM808F.git