Semester Report of Special Problem

Sicong Jiang

December 11, 2019

Summary

This semester I feel fortunate to join the navigation group in IVALab. I'll start with a brief review of this semester's work. In the first two months, I mainly focused on the learning of ROS and Gazebo systems(including the basic commands and operations on ROS,Gazebo and STDR, the sensing parts of turtlebot and some basic navigation knowledge of robotics). Then I began to learn about robot-navigation. I started to learn some classic motion-planning methods like Artificial Potential Field(APF), Virtual Force Field(VFF), Vector Field Histogram(VFH), Reciprocal Velocity Obstacles(RVO) and the DWA/TEB local planner in ROS. Afterward, I became very interested in the navigation and obstacle-avoidance methods of multi-agent system. I applied the Optimal Reciprocal Collision Avoidance(ORCA) and the Deep Deterministic Policy Gradient(DDPG) algorithms to multi-agent system and set up some experiments to test their performance.

1 Description of Works and Codes

1.1 The Learning Module of ROS and Gazebo

Github repositories
`https://github.com/JohnsonJiang1996/Special-Problem/tree/master/Turtlebot_basic`

In this part, I learned the basic structure of ROS system. Including the notion of ROS package, how to create and build a package and how to create a publisher and subscriber in ROS system.

In Gazebo system, I learned how to make a turtlebot complete its own task. And it can be devided into:

1. Moving the turtlebot.
2. Sensing the turtlebot.
3. Wandering with the turtlebot.
4. Sensing the world.

In the first part: moving the turtlebot, I modified the code of goforward.py and drawingasquare.py to change their velocity and angular.

In the second part: sensing the turtlebot, the code kobuki-bumpers-check.py realize the function that checks the status of bumpers. goforward-check-condition.py can make the turtlebot go straight and check the condition of bumper and wheel to decide when to stop. draw-square.py used the gyro information to make the turtlebot rotate as close to 90 degrees as possible.

In the third part: wandering with the turtlebot, the wanderer.py make the turtlebot to explore the surrounding environment and use the data of laser-scan to avoid obstacles just like a wandering zombie.

In the fourth part: sensing the world, kinect.py uses the kinect on turtlebot to show the image of the environment. And the kinect-detection.py catches the object with a specific color in kinect's image. catch-depth.py uses the depth information of kinect to follow an object or person and I modified some of them to realize some specific functions.

What's more, I learned the some basic slam and navigation methods in ROS such as gmapping and AMCL.

1.2 The APF, VFF, RVO and ORCA Algorithms

After finishing the learning module of ROS and Gazebo, I began to learn some classic obstacle-avoidance algorithm which can be applied to the robo-navigation system.

Here is my understanding of some of them.

1.2.1 Artificial Potential Field(APF)

In my understanding, the main features of the APF algorithm are as follows:

1. The robot movement in the environment is considered as a kind of robot movement in the virtual artificial force field.

2. The goal has an attraction and the obstacle has a repulsion.And the force of attraction can be written as the square of the distance between the robot and the goal with a coefficient:

$$U_{att}(\mathbf{x}) = \frac{1}{2}k_a \left|\mathbf{x} - \mathbf{x}_{\mathrm{d}}\right|^2 \tag{1}$$

Similarly, the force of repulsion can be written as:

$$U_{rep}(\mathbf{x}) = \begin{cases} \frac{1}{2}k_r \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 & if \rho \le \rho_0 \\ 0 & if \rho > \rho_0 \end{cases} \tag{2}$$

The final field is the superposition of attraction and repulsion.

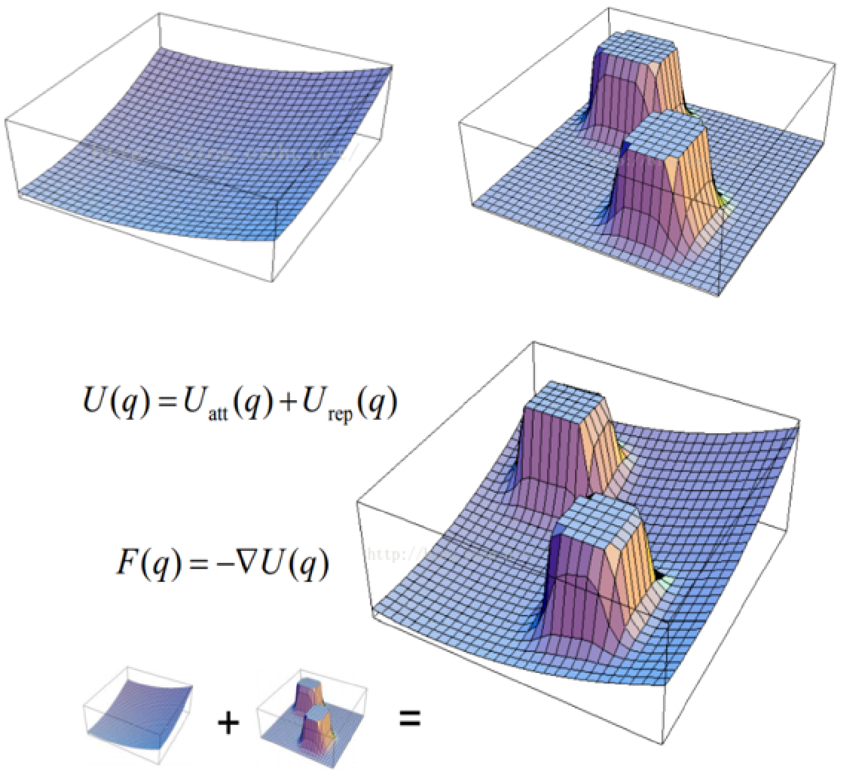


Figure 1: The Final Field

3. Advantages of APF:

- Brief structure.

- Easy to control the robot.
- Can be easily applied to robot in local planning.

4. Disadvantages of APF:
   - Can be trapped in local minimal where the attraction and the repulsion cancel each other. So the robot may not be able to have a direction to go.
   - Oscillations in the presence of obstacles.
   - Sometimes APF gives no passing way between obstacles.

1.2.2 Virtual Force Field(VFF)

In my understanding, the main features of the APF algorithm are as follows:

1. VFF has a 2-d histogram grid, and each cell holds a value which means the confidence of obstacle.

2. VFF measures only one cell at one time and the robot keeps moving.

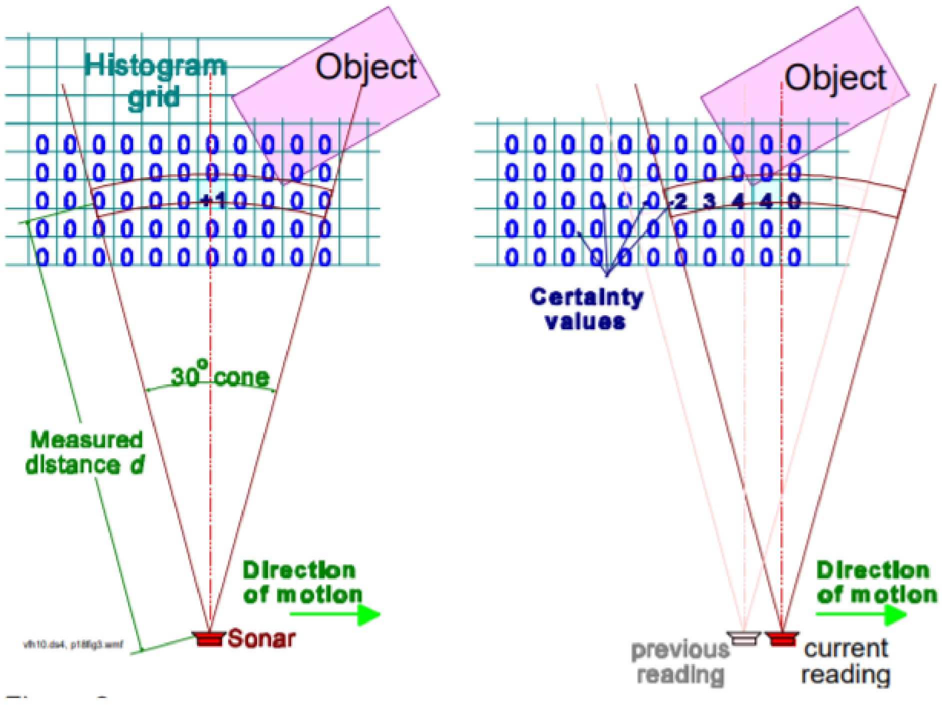


Figure 2: Description of VFF measurements

1.2.3 Reciprocal Velocity Obstacles(RVO) and Optimal Reciprocal Collision Avoidance(ORCA)

Github repositories
`https://github.com/JohnsonJiang1996/Python-RVO2`
(The ORCA appilication and experiments, which is modified from the source code of RVO)

RVO and ORCA are two classic algorithms which can be used as obstacle-avoidance algorithms in multi-agent system.

I applied them to a simple multi-agent system and set some obstacles and navigation goals for agents to test their performance. The experiments can be found in Github(example.py and example2.py). And the introduction and experiment video can be found here.

`https://github.com/JohnsonJiang1996/Special-Problem/tree/master/RVO_ORCA_obstacle_avoidance`

1.3 Some Applications in Multi-agent System

1.3.1 The Learning of STDR

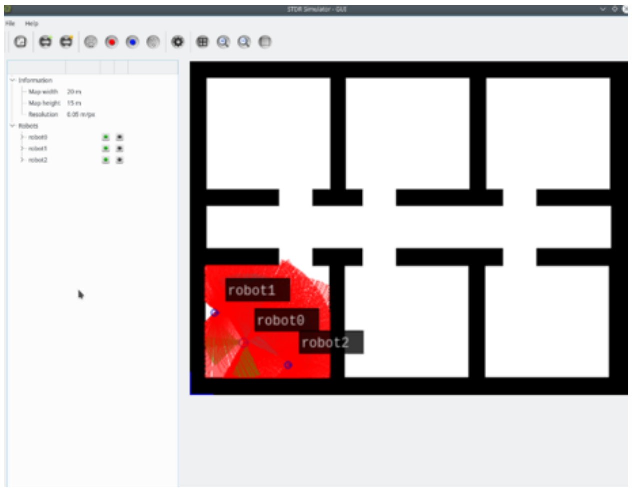
Github repositories
`https://github.com/JohnsonJiang1996/Special-Problem/tree/master/STRD`

Simple Two Dimensional Robot Simulator(STDR) is an easy-to-use robot simulation platform. Its advantage over Gazebo is that I can easily add and control multi-robots in STDR.

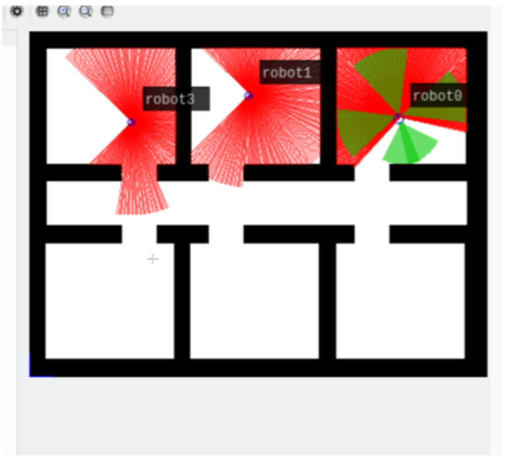
Most modules can be found in src. Some of them are provided by STDR itself and can be easily applied to multi-agent simulation in STDR. I test some of the functions of them such as navigation and SLAM module for multi-agent simulation.

Here are some of my attempts at multi-agent SLAM(The codes are in the repositories above).

1. Set 3 robots in STDR which start from the same place.



(a) Creating 3 robots



(b) Seting the goals of 3 robots

Figure 3: Starting and ending points of robots

2. Using Gmapping method for each agent to make a map
   - Let robot3 use gmapping to explore the whole map and before reaching the goal point.
   - Let robot1 and robot2 find the shortest way to get to the targets.
   - Using ACML to navigate to each point and get the map.
   - Using a simple way to avoid obstacles(a obstacles avoidance sample in STDR, just use the data from laser scan to adjust the angular and speed of robot and to choose the “safe” velocity) .

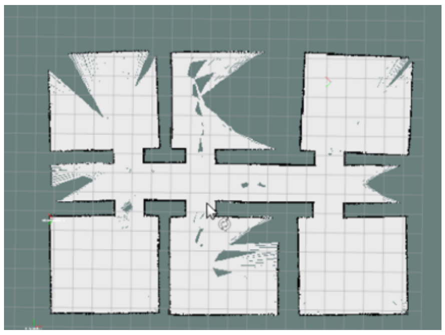
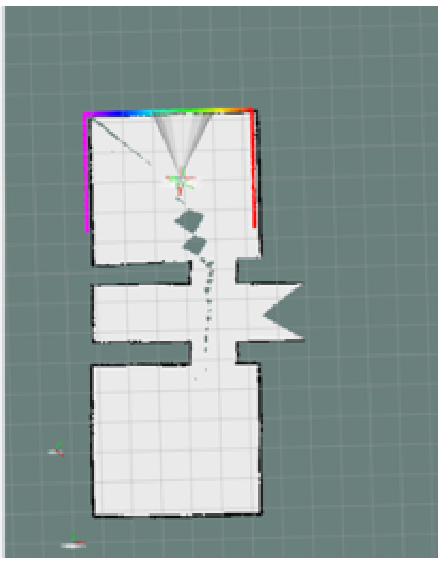


Figure 4: Map created by robot3(Using 162 seconds)

Also, I can get the map created by the three robots.


(a) Map created by robot1(Using 25 seconds)

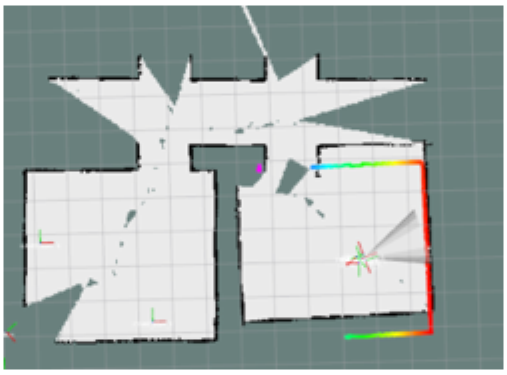

(b) Map created by robot2(Using 27 seconds)

Figure 5: Maps created by robot1 and robot2

From the Figure5 I can see that robot1 and robot2 each creates 1/3 of the whole map created by robot3 while they just spend about 1/6 of the time of robot3. Multi-agent SLAM is more efficient but the difficulty is to combine the maps created by different robots. That is what I'm planning to do in the winter break.

1.3.2 Using DDPG to Train A Multi-agent System To Navigate

Github repositories
`https://github.com/JohnsonJiang1996/Special-Problem/tree/master/DDPG`
(Including the codes I use to train 4 agents in Robotarium to realize simple navigation)

During the study of the course, I came into contact with the knowledge of using deep reinforcement learning to realize robot control. After reading some related paper, I found DDPG can be a good network model to train the multi-agent system to learn how to navigate and avoid obstacles.

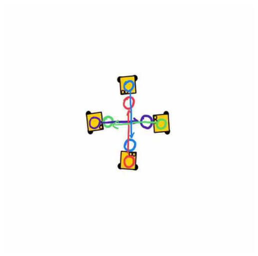
Since I've done some experiments in multi-agent system with ORCA algorithm. So I think trying some learning-based algorithm on multi-agent system and make some comparisons between different algorithms may be interesting. That's why I want to apply DDPG to multi-agent navigation.

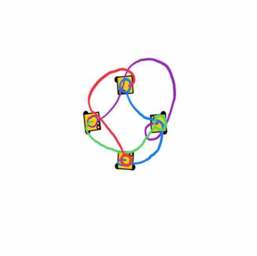
The introduction and structure of DDPG(and Multi-agent DDPG) is here.
`https://github.com/JohnsonJiang1996/Special-Problem/blob/master/DDPG-MADDPG.pdf`

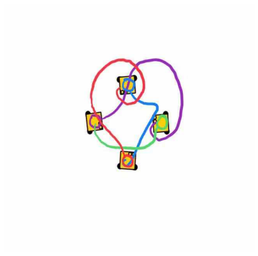
Actually, that's what OpenAI has done(They used an advanced DDPG network called Multi-agent DDPG). So I modify some of their codes to use them in Robotarium environment. Here are some experiments and the results.

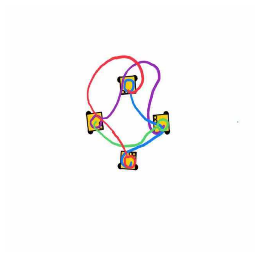
After training of 2200 episodes, I got the results shown in Figure 6 (b),(c),(d),(e). The moving trajectories of each agent were depicted in lines of different color. The simulation process were recorded in a video, where all the agents could move towards the target point successfully without any collision.

In Figure 7 (a),(b), the rewards of one episode grow larger with the passage of time, and get stable in the end. Also, the collision rate goes down to 0 finally.


(a) Agents and Goals


(b) Simulation 1


(c) Simulation 2


(d) Simulation 3

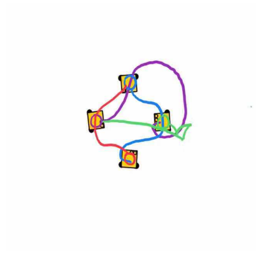
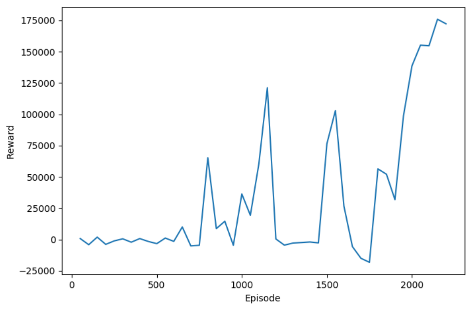
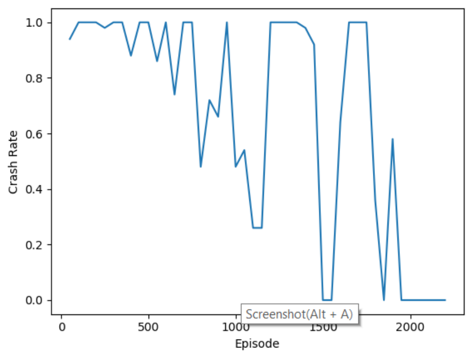

(e) Simulation 4

Figure 6: Simulation of four agents



(a) Rewards



(b) Crash Rate

Figure 7: Rewards and Crash Rate

Drawbacks of DDPG:

By using DDPG, I could successfully accomplish the goal of fixed point control and collision avoidance. However, the case I have studied on is not that complicated. The position of the agents are fixed in the beginning rather than randomly distributed and I only conducted fixed target control rather than more complex control protocols. In terms of the performance of this experiment, as is shown in Figure 2, there are huge flunctuations during the learning process, which might take it long to reach higher reward. More importantly, the uncertainty in the training process would also have negative influence on the convergence of final results in the long term. Actually, in this experiment, it is likely that the reward drops after it gets 'stable' for a short time.

The limitations of DDPG might be caused by changing environment and the uncertainty of other agents' actions. DDPG could definitely behave well while the environment is stable, in this experiment, however, the agents have much 'misunderstanding' about the current situation. That's to say, the agents cannot adjust to the changing environment quickly. To enhance the performance of original DDPG algorithm, I have to take other agents' actions into account and make certain prediction of others' actions, which is the core idea of MADDPG.

However, I haven't modified the code of Multi-agent DDPG to apply it to the Robotarium/ROS environment. So I can just train agents in GYM environment, which is quite different from the real robot simulation. Here are some experiments and results.

MADDPG Experiment Setup:

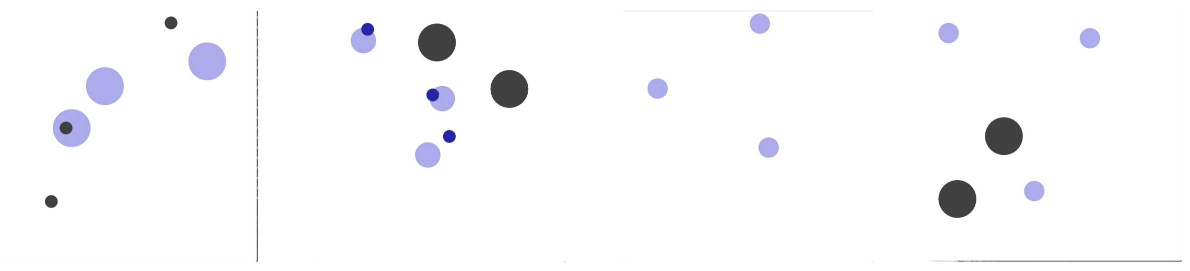

(a) Task 1 (b) Task 2 (c) Task 3 (d) Task 4

Figure 8: Experiment environment for the tasks

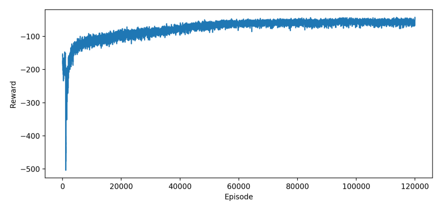
Here I leveraged the multi-agent environment engine from OpenAI. I created our own task scenario as shown in Figure 8. In Figure 8 (a), the blue ones are agents, and the black ones are landmarks. Figure 8 (b), the big blue ones are agents, the small ones are landmarks, and the black ones are obstacles. In Figure 8 (c), the blue ones are agents. In Figure 8 (d), the blue ones are agents and the black ones are obstacles.

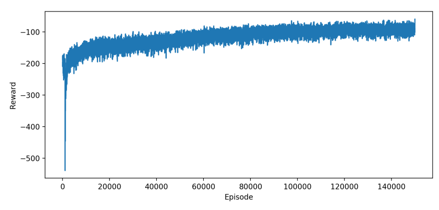
Here I chose three agents and two obstacles because they can well represented similar problems in a more understandable way, and they only require acceptable computational resources for a short-term project. Based on the reward function I designed and the MADDPG algorithm, I trained models for each task on a Macbook Pro with the cpu setting.

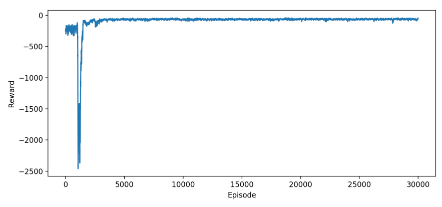
Results and Analysis of MADDPG:

Figure 9 shows the rewards during training for each task. Task 1 was trained for 120000 episodes and the rewards were going up obviously. Task 2 was trained for 150000 episodes and the rewards also had a remarkable upwards trend. However, compared to rewards for task 1 training, those for task 2 developed with more fluctuations, and this should be due to the more complex environment and huger state space in task 2. For task 3 and task 4, both curves had deep troughs at the start of the training, and then grew slowly afterwards. The growing trends for task 3 stopped at an early stage and converged. However, I found that the policy for task 3 might converge sub-optimally, and I would like to explore the reasons in our future work.

As can be seen from Figure 10 (a) and (b), the learned policy for task 1 enables the agent to reach the goal very close, whereas the learned policy for task 2 is not that perfect. In Figure 10 (c), the blue line and orange line are consistent with the red lines after about 40 steps, while the red line has an obvious deviation


(a) Rewards for task 1


(b) Rewards for task 2


(c) Rewards for task 3

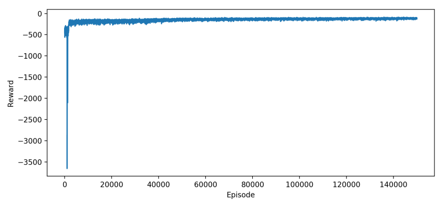
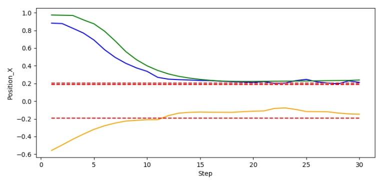
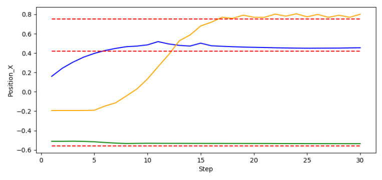
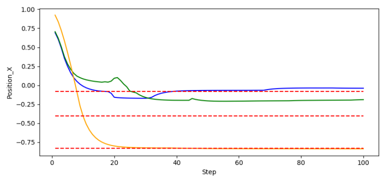

(d) Rewards for task 4

Figure 9: Rewards during training for the 4 tasks


(a) Situation 1 for task 1


(b) Situation 2 for task 1


(c) Situation 1 for task 2

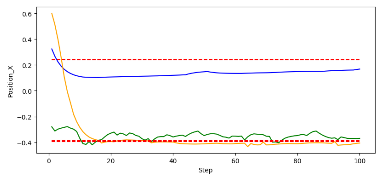

(d) Situation 2 for task 2

Figure 10: The trends of X positions of agents in one execution of learned policy for task 1 and 2. Red dash lines represent the target positions and the solid lines represent the X position of each agent.

from the target. In the case depicted in Figure 10 (d), the green one is more close to its own target than the blue one, but it oscillates around the red line.

That is everything I've done in this semester. And then I'll talk about my expectations and my plans.

2 Two Major Points I Learnt

2.1 First: Try to Understand How Robots Sense the Environment

We have two eyes and use those eyes to collect light that reflects off the objects around us. Our eyes convert that light it into electrical signals that are sent down our optic nerves, which are immediately processed by our brain.

That whole activity, from collecting the light in our eyes, to having an understanding of the world around us, is what is meant by seeing.

However, for robot, the world it feels is quite different from our human beings. We learn the knowledge of our environment from direct image that collected by eyes. But robot need a lot of sensors(Sensors can be laser, camera, or something else). The information it receives from the environment is just data. That's why they need algorithm. Algorithms are like teachers. They teach the machine how to use the data it collects.

At one group meeting, you mentioned that I can try to make a map which is "Robot-friendly" rather than "Human-friendly". I think the reason behind this is that people don't perceive the world the same way robots do. So in order to do a good job with robots, I have to understand how should robot feel the world.

2.2 Second: Discussion with Other Group Members Can be Very Beneficial

Since I'm a beginner in the motion-planning and robot-navigation. I often encounter problems that seems very difficult for me both in methods and practice.

Fortunately, there are so many experienced and excellent members in the group. From the discussion with them, I began to understand some better ideas and methods for doing scientific research.

I feel very glad to participate in the weekly group meeting and have further discussion on their research after the meeting. Although there is still a big gap between my ability and the other members of the group at present, but they always teach me the right way to study robotics.

3 My Expectation at the Beginning

At the beginning of this semester, I expected myself to gain two main abilities during this semester:

Expectation 1.Get familiar with robot operating system and researching methods of robotics.

Expectation 2.After reading a lot of papers, I can create some ideas and start my project.

Under the guidance of the professor and after a semester of study, I think I achieved expectation 1. Now I get to know some basics knowledge of ROS and robotics. Also, I made some small attempts in applying some algorithms in robots.

However, I think I'm still a long way from expectation 2. Now I'm doing some research on multi-agent system. But my understanding of the research status in this field is not comprehensive enough. That's what I still need to do in the future.

4 My Future Work

At the end the this semester, I began to do some research on the multi-agent system. And I found it very interesting. So I plan to do some further research on it during the winter break and next semester. Here are some main points that I need to do :

1. I have applied DDPG algorithm to the Robotarium, but each robot in Robotarium doesn't have many sensors, which makes it hard for them to avoid random obstacles in the environment. So the next step is to apply DDPG and Multi-agent DDPG algorithms to STDR or Gazebo environment. An important advantage of doing so is that in STDR/Gazebo I can use the sensor information(such as laser-scan) as the input information of the model and to train the robots how to avoid obstacles according to the data collected from the laser-scan or camera. I think this may increase the performance and navigation accuracy of robots in multi-agent system.

2. Also, I found that multi-agent system can be used for SLAM. As I presented before, the multi-agent system can create map more effective than single robot. The difficulty is how to combine different maps or different information collected by each robot. In addition, how to judge the quality of the map is also a question. If the map is used for robots to complete some tasks, then it should be easy for the robots to get information. And if map are created for humans, then they should be more intuitive and easy to understand.These are some questions worth my thinking and exploring.

5 How Can I Do Better

Looking back on this semester, I think I should have done better in the following areas:

1. I should improve my ability to substantiate an idea through action. This semester I put too much emphasis on ideas and sometimes too little on action. And I often underestimate the time it takes to implement an idea. As a result, my actual work is not good enough. I should really take a warning and change my mind in the future.

2. I should spend more time on reading excellent papers in the field. Limited literature reading leads to narrow-mindedness in research. I think this semester I haven't read enough papers. Before starting a new project, I should have a comprehensive understanding of how does the problem come up, what is its mathematical model, how many solutions are there, what is the current state, and what are the remaining difficulties... That will help a lot in doing my project.

3. I should do more small projects to become more proficient with ROS. Although I have finished some of the tutorial of ROS, I still find it hard when I try to do some real project with ROS(Such as using DDPG model to train a turtlebot in Gazebo or in STDR). I think this is due to my lack of practical project experience. ROSwiki and coursera are all good source to learn more knowledge of ROS. I should make the most of them in the future.

6 Acknowledgement

I would like to express my special thanks of gratitude to professor Patricio Antonio Vela as well as other members in the navigation group(Shiyu, Ruoyang, Jianning, Fanzhe, Justin and James) who gave me the opportunity and helped me to do some exciting research on robotics. It has been a rewarding semester and I hope I can still do research here during the winter break and next semester.