CPSC 8810
Motion Planning

Assigned: Jan 15, 2020
Due: 11:59pm, Jan 26, 2020

Programming Assignment 1 — Sampling-Based Local Navigation *(Grading: 0–10 points)*

Before starting the assignment, please follow the instructions on the website to set up Python on your personal machine. All projects for this class use Python 3.6.

Problem Description

In the first programming assignment, you will implement the sampling-based velocity approach for local navigation that we covered in lecture 3. In the zipped file `vo.zip` you will find sample code that you need to modify, and three multi-agent navigation scenarios that you can use to test your code. The format of each scenario is as follows:

line 1: the parameters of the first agent:
agent_id, group_id, start_position_x, start_position_y, goal_position_x, goal_position_y, preferred_speed, maximum_speed, radius
line 2: the parameters of the second agent
...
line n: the parameters of the n$^{\text{th}}$ agent

After downloading the code, you should be able to To run the code, you need to type the following at the command line: `python simulator.py`. The file provides a simple `Python` framework for multi-agent navigation that reads scenarios, visualizes the agents, and exports a simulation to a csv file. You should spend some time to get yourself familiar with how the code works. The `agent.py` file contains the `Agent` class that handles the behavior of each agent. Your task is to update this class to enable safe and goal-directed navigation for the agents.

Basic Requirements

Your program should work as follows. It should first load a simulation scenario and set the simulation time step, Δt, to 0.1 s by modifying the corresponding parameters in `simulator.py`. Then, for each simulation step, for each of the n agents you need to compute a new velocity by modifying the `computeNewVelocity` function in `agent.py` as follows:

1. Determine all neighbors that are less than d_H m away from the agent. A typical value for the sensing radius, d_H, to consider is 5-10 m.

2. Uniformly sample N velocities from the admissible velocity space, AV, of the agent. The agent is subject to a maximum speed, so AV is a disc centered at (0,0) having radius equal to the agent's maximum speed. Try different number of candidate velocities between 100 and 1000 in order to find a balance between acceptable navigation behavior and computational performance.

3. Evaluate the fitness of each of the N candidate velocities using the following cost function:

$$\alpha\|\mathbf{v}^{vcand} - \mathbf{v}^{goal}\| + \beta\|\mathbf{v}^{vcand} - \mathbf{v}\| + \frac{\gamma}{tc}.$$

This is a modified version of the function covered in class that accounts for the distance between the candidate and goal velocity, the distance between the candidate and the agent's current velocity, and the risk of the agent to collide with nearby neighbors. Here, α, β, γ are scaling constants that control the relative importance of the three cost terms. You should experiment with different scaling values starting with $\alpha = 1$, $\beta = 1$, and $\gamma = 2$. Regarding the collision cost term, tc denotes the minimum time that it will take for the agent to collide with any of its sensed neighbors if it moves at velocity $\mathbf{v}^{vcand}$. To compute tc, you must first compute the time to collision, τ, between the agent and each of its neighbors as discussed in lecture 2 (though, you should use the candidate rather than the agent's actual velocity).

4. Select as new velocity for the agent the candidate velocity that has the minimum cost. The code in the `update` function will use the selected velocity to update the velocity and position of the agent. It also computes the goal velocity of the agent for the next simulation step by simply taking the unit vector pointing from the current position of the agent to its goal position scaled by the agent's preferred speed.

5. In case an agent is close to its goal (e.g., less than 1 m), you should stop computing new velocities and updating the agent's position, as well as considering this agent for the nearest neighbor computations of the other agents.

Please test your code on the provided `3_agents` and `8_agents` scenarios. Once you are happy with your results, you should export each simulation into a `csv` file by setting the `doExport` flag to `True` in `simulator.py`. You should also document the parameters that you used for each simulation in a `Readme` file.

Extra Credits (4pts)

Besides the 3-agents and 8-agents scenarios, feel free to create and play with your own scenario files. You should also test the `crossing_agents` scenario which is included in the zip file. Keep in mind, though, that each simulation step may be very slow to compute due to the large number of interacting agents in this scenario, especially if you are using a large number, N, of candidate velocities. To address this issue, you may want to set $N = 200$, and also consider only the k-nearest neighbors within the sensing radius of an agent, where k is a small number (2-4 neighbors).

You are also welcome to devise you own cost function. Alternatively, you can account for reciprocity among the agents by following the Reciprocal Velocity Obstacles approach proposed by van den Berg et al. available here (see Section V).

Submission

Submit the assignment using Canvas. You can work in pairs if you want to. If you do so, though, please let us know in advance. Along with your code, please upload the two csv files corresponding to the `3_agents` and `8_agents` scenarios that you simulated, as well as a `Readme` file indicating the parameters that you end up using to simulate each scenario. You're welcome to upload results regarding the `crossing_agents` scenario or any other scenario that you tried, as well as any other file related to the Extra Credits section. Since we are using multiple files, please make a directory and add all files to it. You should zip your directory and upload it to the submission system.

Help

If you get stuck, please do not hesitate to contact us for help, and stop by during the office hours. We also encourage you to post questions and initiate discussions on Canvas. Your colleagues are also there to help you.