Assignment-1: Deadlock Avoidance

COMP 766 / ECSE 683

Deadline: Oct 22, 2020, 11:59pm

1 Objective

Potential field provides a solution for obstacle avoidance, but it is subject to deadlocks when the state, the target, and the obstacle are aligned. In this assignment, you will design/implement a collision-free and deadlock-free path planning method.

Robot

You are free to choose any robot and any simulator for this assignment. You can choose the stardard ones, (Matlab, PyBullet, Drake, ROS) or other simulator that you like. You can use any robot drones, mobile robots, arms, legged robots, etc) from your simulator or create one of your own.

Planner

Make a motion planner that can plan a path from the initial position $\mathbf{x}^s$ to the target position $\mathbf{x}^t$ while avoiding the obstacle $\mathbf{x}^o$ between them.

If you use the original potential field described in the class, you should run into the deadlock. Your task is to create a planner so that this deadlock can be avoid.

You can use anything covered in the course (trajectory generation, potential field, learning from demonstration, trajectory optimization, etc). You can also manually design a path if you think it is a reasonable solution. **There is no wrong answer as long as it works.**

Controller

Take a controller that can read your planned path and send control commands to the robot. You can control with either kinematics or dynamics.

Experimental setup

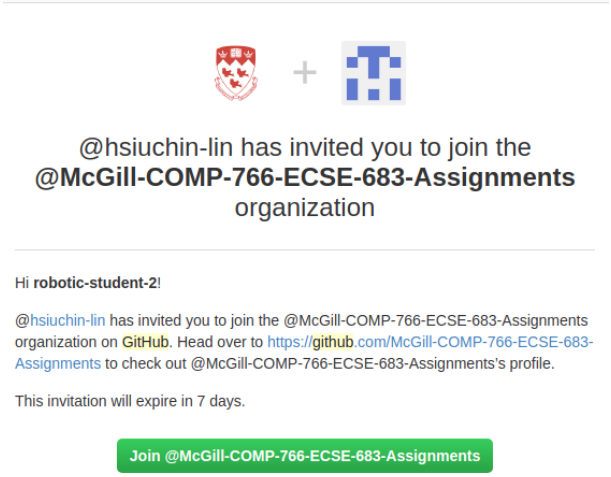
To verify your implementation, you can generate some initial position $\mathbf{x}^s$ and target position $\mathbf{x}^t$ to verify your approach. You can pick a random $\mathbf{x}^s$ and a random $\mathbf{x}^t$ that

are within a reasonable range of your robot. Then, place an obstacle in the middle of the initial and the target to create a deadlock e.g., $\mathbf{x}^o = \mathbf{x}^s + \frac{1}{2}(\mathbf{x}^t - \mathbf{x}^s)$

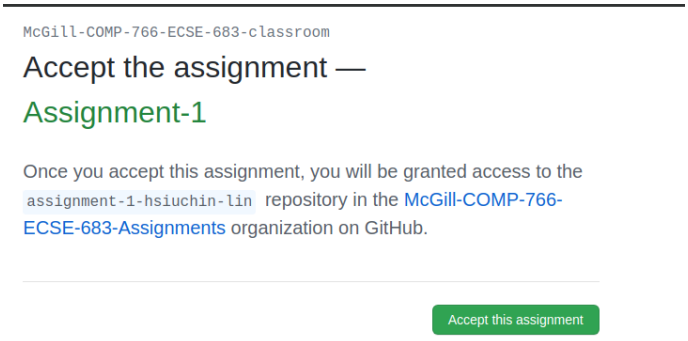
2 How to submit your assignment

Github Classroom

The assignment will be submitted through Github Classroom. If you have never used Github Classroom before, here are some instructions for you. You should have received an invitation to join the `McGill-COMP-766-ECSE-683-Assignments` organization. The email should look like the following.



Once you are part of the organization, please use this link to join the Github Classroom. https://classroom.github.com/a/PunIswNr You should see the following message:



If you click "Accept this assignment", Github Classroom will automatically generate a repository called `assignment-1-[YOUR USER NAME]`. The repository should be empty except a README.md and an example documentation at the top of the directory.

Your repository will be private, and please keep it private. You can use it like a normal github repository and commit as many times as you like. I will check-out your last commit before the deadline.

Required Components

Your submission should include the following items

- source code
- a README.md file that provides the instructions of running your code
- a documentation that describes your solution
- **(optional)** a recorded video of your simulation

FYI: You can use anything from the example code provided in the course, as well as any resources you can find online. However, please provide the references in your documentation.

3 README.md

The README.md should provide instructions to reproduce your experiment and should be at the top of your main directory. You README.md should include, at least, the following information:

- Your name
- List the system requirements if any. If there's no requirement, please specify the environment where the source code is tested, including your
  - Operating system
  - Matlab/Python/ROS and their version number
  - Extra software packages needed
- How to run your source code (which file to run, etc)

4 Documentation

The documentation should describe your examples and experiments. You can include the following components

- Describe your examples

  - What is the configuration space and its degree-of-freedom
  - What is the task space and its degree-of-freedom
  - Describe your state and action
  - Describe your test cases
- Describe your method
  - What is your approach?
  - How do you avoid obstacles?
- Justify your approach. How can you convince others that your method works well? (e.g., show minimum distance from the obstacles, snapshots of your simulator, etc.).
- Discuss any limitations. Does it work 100% of the time? Why?
- Provide references if any. If you take any code online or follow some papers, please cite it.