Motion Planning RBE 550

Project 0: Getting Familiar with OMPL

DUE: Thursday August 29 at 5:00 pm.

All written components should be typeset using LaTeX. A template is provided on Overleaf. However, you are free to use your own format as long as it is in LaTeX.

Submit two files **a)** your code as a zip file and **b)** the written report as a pdf file.

Present only your own work. You must *explain* all of your answers. Answers without explanations will receive no credit. **This assignment must be completed individually and not in pairs.**

Goal of this Project

A core component of this class will be getting familiar with and using the Open Motion Planning Library (OMPL). The goal of this project is to help you install OMPL, understand some of the basic requirements for this course, and become acquainted with the style of the course projects. In this class besides the core Motion Planning Concepts, you will also learn to use, modify, and extend OMPL. You can find more information about OMPL and its documentation on the official webpage https://ompl.kavrakilab.org/. You can also find many useful tutorials on the OMPL Tutorials Page.

1 Installing OMPL

There are different ways to install OMPL. Here, we propose what we believe is the easiest method for each system, but ultimately, it will be up to you to determine the best way to set it up on your system.

1.1 Option1: Install with Docker - Recommended For Windows/Linux Users

For Windows and Linux users, we propose using Docker. All the assignments will be tested in Docker, so it will be easier for you to choose this option to avoid any compilation or compatibility issues. You can learn more about Docker here.

1.1.1 Install Docker

There are two ways to install Docker: installing Docker Engine **OR** installing Docker Desktop. You can read an explanation of the difference here. In a nutshell, Docker Desktop provides a GUI for running Docker commands, while Docker Engine is terminal-only.

1.1.2 Install OMPL inside Docker

After you successfully install Docker, pull the OMPL image for the RBE class. Open a terminal and type:

```
# Pull the image
docker pull cchamzas/rbe550-ompl:initial-commit
# Run the image and mount the project files location to a folder named code
# In my setup, <project-path> is /home/cchamzas/Project0-Install-OMPL/code
docker run -it -v <project-path>:/code/ \
cchamzas/rbe550-ompl:initial-commit /bin/bash
```

The `-v` flag mounts an external folder inside the Docker container. This allows you to modify the project files outside the container and build the project within the container. You are free to use any development environment that suits you.

After executing the last command, you should be inside the Docker container. Navigate to the `/code` folder, where you should see the project files.

1.2 Option2: Installing from Source - Recommended for Mac Users

Unfortunately, Docker might not run on most Mac systems, so you will need to install OMPL from source. Consult the OMPL Installation Page for detailed instructions. If you are a Mac user, you should just run:

```
brew install ompl
```

You are welcome to install from source on other systems, but you need to be comfortable with the process.

2 Build and Run OMPL

Irrespective of which installation process you follow, you can now build and run an OMPL demo. Open a terminal and type the following:

```
cd code
# Build the Point2DPlanning demo.
mkdir build && cd build
cmake ../ && make
# Run the Point2DPlanning demo.
./Point2DPlanning
```

Project Exercises

1. **(10 points)** What method did you use to install OMPL? How long did it take you approximately?
2. **(40 points)** Demonstrate that you have successfully installed OMPL. In your report, include the terminal output after running the `Point2DPlanning.cpp` demo, and also include the `result_demo.ppm` image that was created.
3. **(20 points)** Understand and modify the demo. Skim the source code of `Point2DPlanning.cpp`,

change the start and goal locations of the problem, and include the new `.ppm` image in your report. Try to make it sufficiently different from the previous one.

4. **(20 points)** Find one C++ demo and one Python demo from here, and run them. For the C++ demo, you will need to modify the `CMakeLists.txt` file accordingly. In your report, include the name of the demo you ran and the terminal output. It is possible that some of the more complicated demos do not run, so try to find one that does.
5. **(10 points)** Submit the correct files in the correct location on Canvas. You should submit one PDF with your report typeset in LaTeXand a zip file with your code.
6. **Bonus: (25 points)** Create your own `.ppm` environment and solve a motion planning problem there. You can use maps from the web or draw your own images. Include the path and the created map in your report.

3 Pro Tips

1. As this is more of a setup project, you can collaborate as much as you need and ask any questions on Piazza. However, the final report must be your own work. pliagiarism will not be tolerated.

2. Don't blindly follow the commands outlined here; try to understand what you are doing.

3. Yes, you can use LLMs as tools, but you must be able to explain what you have done and why.

4. If you are using Visual Studio Code, It has a very useful plugin for Docker that allows you edit files easily, and insteract