Project 2: Motion Planning

Collaboration in the sense of discussion is allowed, however, the assignment is individual and the work you do should be entirely your own. See the collaboration and academic integrity statement here: https://natanaso.github.io/ece276b. *Books, websites, and papers may be consulted but not copied from. It is absolutely forbidden to copy or even look at anyone else's code.* ***Please acknowledge in writing people you discuss the problems with.***

Submission

Please submit the following files on **Gradescope** by the deadline shown at the top right corner.

1. **Programming assignment**: upload all code you have written for the project and a README file with a clear, concise description of the main file and how to run your code.
2. **Report**: upload your report in pdf format. You are encouraged but not required to use an IEEE conference template[1] for your report.

Problems

In square brackets are the points assigned to each part.

1. This project will focus on comparing the performance of search-based and sampling-based motion planning algorithms in 3-D Euclidean space. You are provided with a set of 3-D environments described by a rectangular outer boundary and a set of rectangular obstacle blocks (see Fig. 1). Each rectangle is described by a 9 dimensional vector, specifying its lower left corner $(x_{min}, y_{min}, z_{min})$, its upper right corner $(x_{max}, y_{max}, z_{max})$, and its RGB color. The start $\mathbf{x}_s \in \mathbb{R}^3$ and goal $\mathbf{x}_\tau \in \mathbb{R}^3$ coordinates are also specified for each of the available environments. The provided sample code includes a baseline planner which moves greedily towards the goal. This planner gets stuck in complex environments and is not very careful with collision checking.

Part 1: Implement an algorithm for checking collisions between line segments and axis-aligned bounding boxes (AABBs) in continuous 3D space in order to evaluate the safety of a planning algorithm. You are free to use an existing library or implement your own algorithm.

Part 2: Implement your own search-based planning algorithm (e.g., weighted A*, jump point search, etc.) based on the ideas discussed in the lectures. You are encouraged to try different ideas to improve the efficiency and optimality of your algorithm.

Part 3: Download, install, and learn to use one of the following libraries:

- The state-of-the-art sampling-based motion planning library OMPL (https://ompl.kavrakilab.org/). Various demos and tutorials are available on the OMPL website[2]. Two useful examples can be found here: https://ompl.kavrakilab.org/OptimalPlanning_8py_source.html, https://ompl.kavrakilab.org/RandomWalkPlanner_8py_source.html
- The Python motion planning library at https://github.com/motion-planning/rrt-algorithms. Examples are available in the repository.

While OMPL is preferable, it might be challenging to install. You can use either library to implement the RRT or RRT* planning algorithm on the same 3D environments.

Write a project report describing your approach to the 3D motion planning problem. Include the following sections:

(a) [5 pts] **Introduction:** present a brief overview of the motion planning problem and the techniques used in this project.

[1] https://www.ieee.org/conferences_events/conferences/publishing/templates.html
[2] Tutorials: https://ompl.kavrakilab.org/tutorials.html; Demos: https://ompl.kavrakilab.org/group__demos.html

(b) [8 pts] **Problem Statement:** state the problem you are trying to solve in mathematical terms. You should include the basic elements of the determinstic shortest path problem in continuous 3-D space.

(c) [30 pts] **Technical Approach:** describe the search-based planning algorithm that you developed, including key ideas that enable your planner to scale to large maps and compute efficient solutions. You should discuss the properties of your implementation: (sub)optimality, completeness, memory and time efficiency, etc. Describe also the sampling-based algorithm your chose to use from OMPL. The description of the sampling-based algorithm should be more concise since you did not implement it yourself but you should still describe, which algorithm you chose, what parameters you specified, how you implemented any required components, etc.

(d) [27 pts] **Results:** compare the performance of your search-based algorithm and the OMPL sampling-based algorithm in terms of (1) quality of computed paths, (2) number of considered nodes, (3) the effect of different parameters such as choice of heuristic function or choice of sampling, etc. Discuss any interesting details about the performance of the two classes of algorithms. It is understood that since the OMPL algorithm is implemented in C++, it may be more memory and computation efficient than your own implementation but you should still discuss general trends and observations you see. Include images showing the paths (and other characteristics if desired) computed by the algorithms on the various environments.

(e) [30 pts] **Code:** upload your code to Gradescope. It will be evaluated based on correctness and efficiency. The results presented in the report should be consistent with the output of your code. The code should have a clear structure and comments. **Copying, rephrasing, or even looking at anyone else's code is strictly forbidden. Writing your own code is the best way to avoid plagiarism discussions or consequences.**

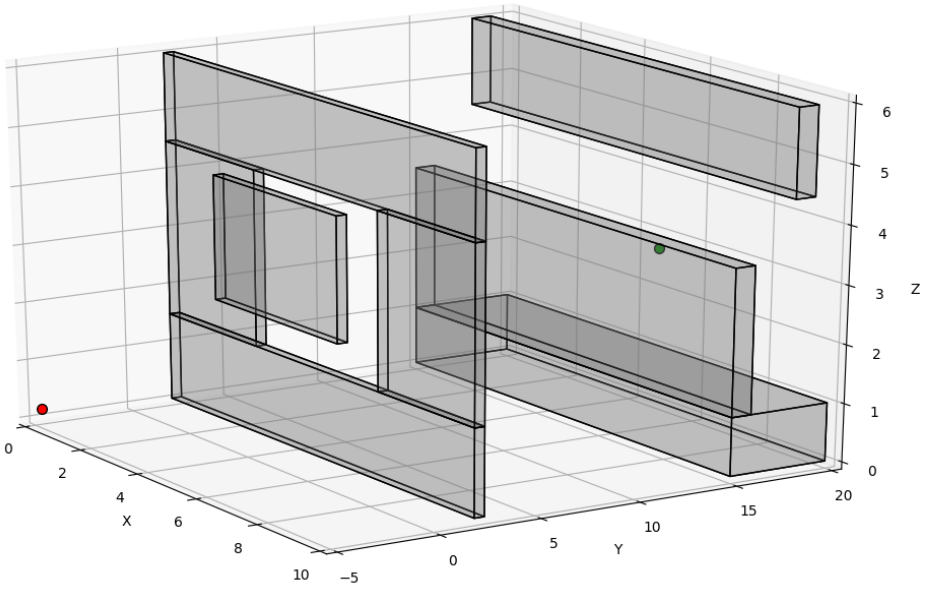

Figure 1: The **Window** planning environment showing the obstacles in gray, the start position in red, and the goal position in green.