EECS 498: Introduction to Algorithmic Robotics
Fall 2020
Final Project
Project Choice Due 11/18/2020 at 11:59pm
Writeup and Demo Due 12/18/2020 at 11:59pm

Rules:

1. **This project must be done either individually or in a group of two people. Each topic specifies the maximum group size**.
2. You are encouraged to post questions on Piazza.
3. No late submissions will be accepted.
4. Submit your code along with a `pdf` of your write up in a zip file to Canvas. Do not paste your code into your `pdf`.
5. Remember that copying-and-pasting code from other sources is not allowed.

The goal of the final project is to give you more in-depth experience with a chosen area of robotics algorithms. The project topics below are chosen so that you have an opportunity to expand on a homework problem or investigate a topic presented in class that was not on the homework. The topics are intentionally open-ended, which means you'll need to come up with not only the solution, but also interesting examples and benchmarks that show your solution works. Feel free to look around the internet for resources to help you implement your project. However, the core code (i.e. everything except helper functions for visualization, data parsing, etc.) should be your own.

Topics

1. **Convex Optimization**. Here you will improve on the Barrier Method from HW2. First, incorporate equality constraints into the Barrier Method from HW2. Second, incorporate a way to find an initial feasible point. Show that the new method works on several 2D examples which include equality constraints. Input a 6-dimensional linear program with both equality and inequality constraints into your method and show that it generates the same answer as an existing solver (e.g. CVX).

   Maximum group size: 1

2. **Search-based Planning**. Starting from your implementations of A* in HW3, implement the ANA* algorithm (link). Use "8-connected" space. Compare ANA* and A* on several interesting navigation problems for the PR2. Come up with your own admissible heuristic and compare it to the Euclidean heuristic using both ANA* and A*. For each problem, generate a graph similar to Figure 3(a) in the paper, showing the solution cost vs. time for ANA* with each heuristic, as well as the solution cost of A* using each heuristic. Set a long timeout for ANA* to ensure you eventually get the optimal path.

   Maximum group size: 1

3. **Kinodynamic RRT**. Implement the Kinodynamic RRT algorithm for a planar hover-craft robot in openrave (should be able to accelerate/decelerate in x, y, and θ). You will need to create the robot model (can be just a box) and write a function to simulate the robot's movement inside the motion planner. Make an interesting environment that contains obstacles. The goal should be a small target region in the environment. Try using different numbers of motion primitives and evaluate performance in terms of path quality and computation time for increasing numbers of primitives. Generate an image showing the search tree produced by the planner in an interesting environment and a video showing the execution of the planned trajectory.

   Maximum group size: 1

4. **Point Cloud Processing**. Implement the Point Feature Histogram method. Use the distance between histogram signatures instead of the distance between points inside ICP to align two point clouds. To reduce computation time, you will need to come up with a method to determine which points should be considered for computing the signatures and which should be ignored. Test varying numbers of bins on aligning two point clouds that partially overlap. Generate some synthetic point clouds to show your method works. Also, get some real-world point clouds from the internet (e.g. here) and show that the method works with them.

   Maximum group size: 2

5. **Localization**. Consider the PR2 robot navigating in an openrave environment with obstacles. Implement a function that simulates a simple location sensor in openrave (i.e. the function should return a slightly noisy estimate of the true location). Pick an interesting path for the robot to execute and estimate the robot's position as it executes the path using a) a Kalman filter, and b) a particle filter. You will need to tune the noise in the sensing and action and the parameters of the algorithms to make sure there is enough noise to make the problem interesting but not too much so that it's impossible to estimate the location. Compare the performance of the two algorithms in terms of accuracy in several interesting scenarios. Produce a case where the Kalman filter is unable to produce a reasonable estimate (e.g. the mean is inside an obstacle) but the particle filter does produce a reasonable estimate. Include the motion model, sensor model, and noise covariances you used in your report.

   Maximum group size: 2

6. **Propose your own topic**. The topic should be relevant to the course material and about the same level of difficulty as the projects above. To propose your topic, fill in the appropriate part of the project choice form (see below). Include an overview of the project and a description of the anticipated tests (similar to the descriptions of the projects above). The professor will then decide whether or not to approve your project.

   Maximum group size: 2

Registering your choice

Once you decide on the project topic and on your group (if applicable), register your choice by filling out this form. **You must fill in this form by November 18th**.

What to Submit

1. (30% of grade) **Demo scripts**. Your submission should include a shell script called `install.sh`, which compiles/installs any required packages using `pip install`. You can assume `openrave`, Python 2.7 or 3 (please specify which you are using in `install.sh`), `numpy`, and `matplotlib` are already installed. You should also have a file called `demo.py`. When we run `demo.py`, your method should run and produce a visual output showing that it has solved an interesting problem. We will only run the commands `./install.sh` and `python demo.py`, not any other commands. `demo.py` should print out an expected time to run (this can be a range). Run time must be less than 30 minutes no matter what project you did. Demos that do not run, exceed the specified time, or crash will receive a grade of 0. Pick your demo example carefully!

2. (70% of grade) **Project Report**. Submit a single-spaced, 11pt font `pdf` report describing what you did for the final project. Your report should be at least four full pages long. Your report should include the following sections:

   - **Introduction**: Motivate why the problem you are solving is important. What applications require this kind of method?
   - **Implementation**: Describe, in detail, using equations and illustrations where necessary, what you did to implement your project. For many projects a block diagram showing the steps involved in the method is very helpful. Make sure to discuss *why* you made the decisions you did.
   - **Results**: First describe, in detail, the experimental setup you used. Then describe each result in sequence. Make sure to discuss the conclusions of each test (e.g. "algorithm a is better than algorithm b in terms of...." or "this test shows that the algorithm can...."). Do not put numbers without explanation.