EE183DA (Winter 2019)

Design of Robotic Systems I

Prof. Ankur Mehta
mehtank@ucla.edu

Lab assignment 4
Due 5pm Thu. Feb. 28, 2019

1 Lab Overview

1.1 Objectives

The goal of this lab is to explore build autonomy into a simple 2 wheeled robot. Using your description of the system dynamics from the previous lab, you will develop an RRT-based planner designed to get your robot to a desired goal state.

1.2 Deliverables

As a team, you will create a well documented git repository containing all your code and data. You will also create a team writeup describing the mathematical formulation, experimental setup, and experimental results and data.

As an individual, you will create your own one-page slide summarizing the key contributions of your work primarily using equations and/or figures. You will be assessed on both the clarity and completeness of your content.

Include a link to your code repository / website in your report, as well as a complete list of references you've used and in what manner, and submit pdfs by 5pm Thu. Feb. 28, 2019.

Submissions that are up to 24 hours late will be accepted for a 10 percentage point reduction in final grade. No submissions will be accepted more than 24 hours late.

2 Lab specification

2.1 Robot model

You will again consider the 2 wheeled robot from labs 1/2. For simplicity when considering collisions, you can consider the boundary of the robot to be a circle; choose a conservative diameter that completely encompasses the robot body. For realism, you can optionally consider the true rectangular shape of the robot.

2.2 Trajectory planning

The robot must move through a cluttered 2D rectangular environment representing a parking lot. You will want to consider geometric obstacles defined by rectangles specified within the space. The robot must achieve a prescribed goal, i.e. a desired x, y, θ for the robot state, within this environment. Note that this robot is non-holonomic: you only have two input controls for the 3-DOF state.

Compute a plan to take the robot from a given initial state to the desired goal state while avoiding obstacles, and use that to implement a planner / controller specifying the inputs to the actuators as a function of time.

2.2(a). Translate the map of your environment (with obstacles) into C-space, and create a visualization of this map with initial and goal states indicated.

2.2(b). Given a set of points V in C-space and a single other target point, write and test a function to determine which of the points in V is closest to the target.

2.2(c). Given arbitrary initial and target robot states (in C-space), write and test a function to generate a smooth achievable trajectory from the inital state towards the target lasting 1 second. What are the control inputs for this trajectory?

2.2(d). Given your C-space map and an arbitrary robot trajectory, write and test a function to determine whether this trajectory is collision free.

2.2(e). Put these functions together to implement an RRT planner on your map to generate a trajectory from a specified initial state to the desired goal state. Visualize the evolution of the RRT as well as the resulting trajectory.

2.2(f). Since your system dynamics are reversible, modify your planner to generate robust trajectories to the goal state from arbitrary initial states.

2.2(g). Optionally, improve on this planner using RRT* or LQR trees.

2.3 Evaluation

You will evaluate your robust planner first in simulation, then implement and test the planner on your paperbots from lab 1. You can either use your Kalman Filter from lab 2, or an external (perhaps camera based) ground truth sensor to generate the state estimate.

2.3(a). Run some examples that demonstrate the performance (in terms of computational efficiency, trajectory efficiency, and obstacle avoidance) of your planner as your robot tries to achieve various goals (such as head-in parking and parallel parking between other such parked vehicles). Clearly describe the experiments that were run, the data that was gathered, and the process by which you use that data to characterize the performance of your planner. Include figures; you may also refer to animations and videos uploaded to your git repo.

2.3(b). How much relative computational cost is associated with the various operations of the RRT planner?

2.3(c). If the obstacles were dynamic, and themselves moved, you would need to re-plan trajectories to account for the varying environment. Based on the computational time of your planner, what obstacle dynamics would you be able to handle in real time?

2.3(d). Qualitatively describe some conclusions about the effectiveness of your planner for potential tasks your robot may encounter. How might you improve it?