Integrated Path and Trajectory Planning System for Autonomous Quadcopter Navigation in Dense Outdoor Urban Environments

Overview

The goal of this independent project was to gain experience with path and trajectory planning algorithms, C++, and to prepare to begin a course of formal study at University of Maryland, College Park: the M.Eng. in Robotics Engineering.

To this end, I have, with this project, developed an integrated path and trajectory planning system in C++ that does well to compute smooth, collision-free, near shortest-distance trajectory plans between any two points on large, obstacle-dense 3D maps of outdoor urban spaces. I designed the planning system to comprise three main subsystems: coarse path planning with A* graph search, refined path planning with RRT, and trajectory planning with a 7th order polynomial solver I built from first principles that does well to approximate minimum snap trajectories.

Performance and Validation

Through Monte Carlo simulations, I have thus far characterized the performance of the system as follows:

- Global Path Planning: <0.2s for target waypoints ~259m apart.
- Local Path Planning: <0.12s for waypoints spaced ~50m apart.
- Trajectory Planning: <0.15s to solve for time-ordered trajectory between ~100 waypoints spaced 1m apart.

Trajectories between points are typified with this example of a plan computed between (0,0,0) and (200,200,200) using a desired average speed of 2 m/s.

As I developed this project, I've considered optimizations, performance tradeoffs, and technical challenges, including the following, which I worked to solve:

- Balancing global planning resolution with execution time -- I solved by finding a middle-ground planning resolution
- Efficiently finding the nearest obstacle to a query point given datasets with >4000 obstacle -- I solved by employing a spatial data structure called a KDTree
- Constructing a trajectory planner the generates smooth trajectories between waypoints -- I solved by building a custom 7th order polynomial interpolation module from first principles

Dependencies

This project depends on a KDTree library called KDTree (by J. Frederico Carvalho) and the Eigen library.

Note

This is an evolving project. I hope to expand on this project this semester in ENPMXXXX (Planning) as a first semester M.Eng. student at University of Maryland, College Park.