MEAM620 Advanced Robotics
Project 3 Report

Akash Sundar

I. Introduction

This project is a culmination of work carried out for the course Advanced Robotics at the University of Pennsylvania. It entails integrating a controller, path planning algorithms (A* and Dijkstra), trajectory generation (minimum snap) and an Error State Kalman Filter (ESKF) implemented in previous projects in order to simulate the flight of an unmanned aerial vehicle (UAV) in various environments. The gains were fine tuned and the velocity was adjusted to account for the introduction of an error to the state of the system. The following sections discuss a brief overview of the individual components that went together to produce a complete Obstacle Avoidance system.

II. Controller

A geometric PD controller was implemented for this assignment. Geometric non-linear controllers are well suited for more aggressive maneuvers, i.e., higher speeds and sharp turns. I intended to make the most of the robust flying action of the Crazyflie and hence used the geometric controller. Given a desired state of the drone, the controller computed two inputs u1 and u2 used to command the thrust and the torque respectively.

The controller was re-tuned for the final project to perform as closely possible to the expected desired state received from the Error State Kalman Filter.

The final set of gains used are as follows:

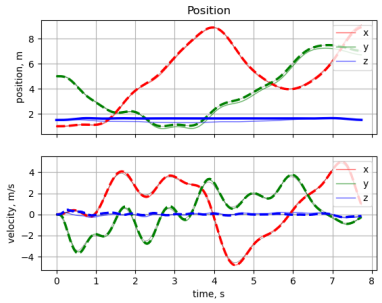
$$K_p = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 20 \end{bmatrix} \ (s^{-2})$$

$$K_d = \begin{bmatrix} 6.2 & 0 & 0 \\ 0 & 6.2 & 0 \\ 0 & 0 & 9 \end{bmatrix} \ (s^{-1})$$

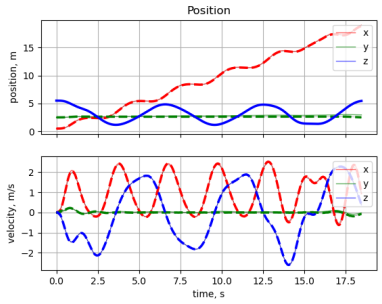
$$K_R = \begin{bmatrix} 2800 & 0 & 0 \\ 0 & 2800 & 0 \\ 0 & 0 & 160 \end{bmatrix} \ (^\circ s^{-2})$$

$$K_\omega = \begin{bmatrix} 125 & 0 & 0 \\ 0 & 125 & 0 \\ 0 & 0 & 75 \end{bmatrix} \ (s^{-1})$$

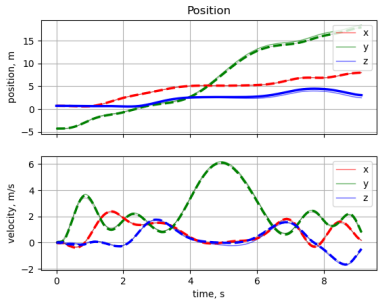
The position vs velocity plots for 3 standard enviroments, namely, Maze, Over and Under and Window have been plotted in Figure 1.



(a) Maze



(b) Over Under



(c) Window

Fig. 1: Position vs Velocity

III. Trajectory Generation

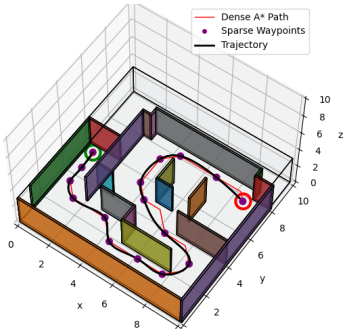
In order to determine an optimum obstacle free path for the drone given a known environment, graph search planning, I first implemented an A* Algorithm that aims to find the shortest path between the start node and the end node owing to the fact that it uses a heuristic to guide our search. A heuristic is a function that receives a state and outputs an estimate of the optimal path's cost from to the closest goal.

A map resolution of 0.25 and a margin of 0.5 around the obstacles was used to create the OCC map to run the path planning algorithm on. The returned path was a set of dense waypoints that needed to be pruned to reduce complexity and produce smoother control.

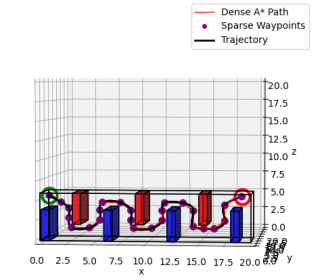
I performed pruning using RDP algorithm, wherein I identified the points of maximum inflection or change in direction and marked them as my primary waypoints in my newly generated path. I also used the example of the window trajectory to imagine the result of such an algorithm in a long and narrow passage. An optimal spline in such a passage might overshoot the maximum

height of the space and hence I added a secondary degree of control by adding waypoints in the midpoints of every one of my primary waypoints.

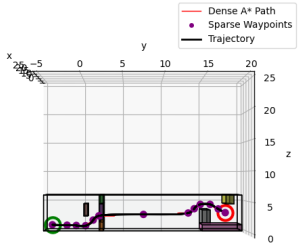
In an attempt to reduce sharp turns and provide allow for more aggressive maneuvers, I fit a spline over my pruned set of waypoints. I implemented minimum snap, minimum jerk and constant speed trajectories to account for this problem. Constant speed trajectory simply fit straight lines between the points and was hence the least effective among the three. I successfully fine tuned the gains for minimum snap trajectory and navigated the drone safely through the obstacle course environments. I also increased the duration of the first and last segment by a factor of 2 to allow for sufficient time for the drone to get a sufficient reading of its environment for better control and to reduce obstacle collisions on random seeds. Figure 2 represents the trajectory generated for the 3 maps.



(a) Maze



(b) Over Under



(c) Window

Fig. 2: Trajectory of Quadrotor

IV. Error State Kalman Filter

An Error State Kalman Filter (ESKF) was implemented in order to update the state of the drone based on the measurement received and a probabilistic idea of the state. This is a variation from previous assignments as it assumes an inherent inaccuracy in measurement coupled with noise in the measurement of states of the accelerometer and gyroscope. The VIO odometry makes use of features from the obstacles and boundaries which are projected into the camera frame of the quadrotor and also takes into consideration random noise caught by the sensors for a more realistic representation of the environment. The singular trace of the covariance matrix provides a measure of information available for the system as seen in Figure 3. To avoid repetition, only one of the graphs pertaining to the Maze environment has been added, although the remaining plots are practically identical save for the scale of the x axis which is a measure of the shape of the covariance matrix for that environment.

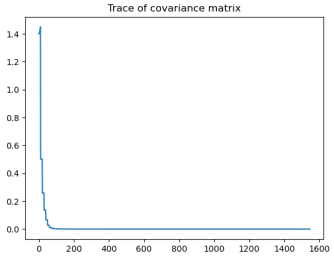


Fig. 3: Trace of Covariance Matrix

V. Results

A initial set of gains as implemented in Proj 1.3 could be implemented with a velocity of 10-12 m/s. However, a probabilistic error introduced into the measurement of state of the quadrotor severly limited the velocity to a maximum of 4 m/s. The following table represents the effect of velocity in the Window environment.

Parameters	v = 10 m/s	v = 4 m/s
No Collision:	FAIL	PASS
Stopped at Goal:	FAIL	PASS
Flight Time:	5.8 seconds	11.1 seconds
Flight Distance:	44.3 metres	27.1 metres
Planning Time:	51.1 seconds	51.8 seconds

Figure 4 shows the optimal trajectory as followed for Maze environment. Again, to avoid repetition. the plots for the remaining environments have not been enclosed. However, they follow the planned trajectory very closely.

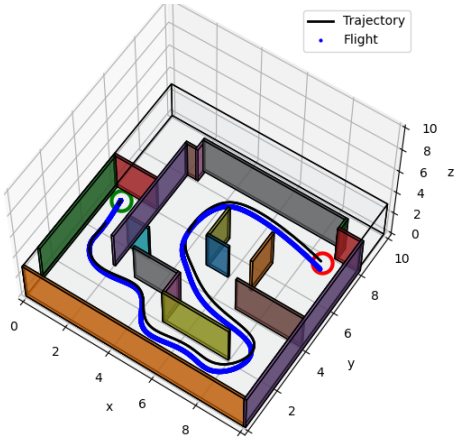


Fig. 4: Flight Path