MEAM 620 Project 3 Report

Lenning Davis

I. Introduction and Approach Overview

The goal of this project is to fly a simulated quadcopter through a known environment in the shortest time possible. This involves searching for a path, planning a trajectory, tracking the trajectory, and localizing the quadcopter position. In developing my solution to this project, I chose to build upon the code that I had written and rigorously tested for previous projects.

To search for a path I use A* with an euclidean distance heuristic. Before solving for a trajectory the Ramer–Douglas–Peucker (RDP) algorithm [1], [3] is used to remove excess points along the path. Next, a minimum jerk trajectory is solved to fit the sparse waypoints [2]. The solved trajectory is checked against the global map to find collisions of the planned trajectory with obstacles. If a collision is found, an extra waypoint is added along the A* path at the midpoint between the two previous trajectory waypoints. This process continues iteratively until a trajectory is found without collisions. The quadcopter then follows the trajectory with the same SE3 control algorithm as developed in project 1. The quadcopter localizes its position with the same visual odometry algorithm as developed in project 2.

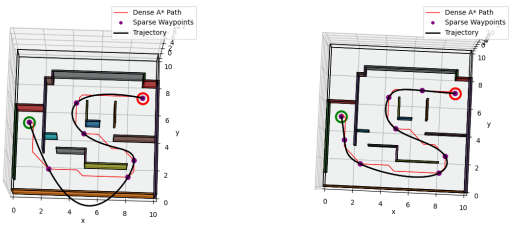
II. Encountered Issues and Improvements on Projects 1 & 2

In adapting my code from projects 1 and 2 for project 3, I identified several major problems that motivated changes to my approach.

A. Covariance Increase and Localization Drift

One of the major differences between project 1 and project 3 is the issue of state estimation using Visual Inertial Odometry (VIO). In project 1 the true position of the quadcopter is known, thus the SE3 controller can drive the quadcopter from where it currently is (true state) to where it needs to go (desired state). However, in project 3 the SE3 controller can only drive the quadcopter from its *estimated state* to where it needs to go (desired state). When I first implemented the old code, I found that the quadcopter tracked the desired trajectory exactly, but was not arriving at the goal waypoint; the estimated position was slowly drifting away from true position of the quadcopter. The estimated state of the quadcopter is calculated using an Error State Kalman Filter (ESKF) receiving accelerometer and vision updates, thus the error state covariance is dependent upon the magnitude of commanded accelerations for the quadcopter. With this observation and conceptual understanding, I adapted my code to reduce the magnitudes of acceleration in planning and execution. This was achieved by implementing an iterative minimum jerk planner and increasing the time for the first segment.

The planner I had for project 1 did not perform well when the true position of the quad was not known. Large commanded accelerations caused by deviations from a straight line trajectory lead to drift in state estimation towards the end of the flight. To amend this issue, I first implemented a minimum jerk planner. I found that the minimum jerk planner did not work well with large numbers of waypoints; when waypoints were close together the planner would still return large accelerations. To amend this issue, I pruned the trajectory aggressively using RDP [3], [1], then ran the planner through multiple iterations - adding more points each loop until a trajectory is found without collisions. Shown in Fig. 1, the result is a trajectory with as few waypoints as possible.



(a) Single Iteration (b) 3 Iterations

Fig. 1: Iterative minimum jerk trajectory planner example. Note the fixed collisions from Fig. 1a to Fig. 1b.

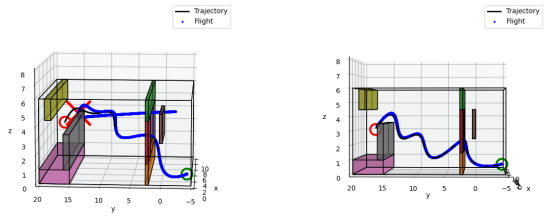
In testing, I also found that the time to arrive at the initial waypoint had a large impact on the state estimation drift towards the end of the flight. As mentioned in the previous paragraph, the waypoint times are calculated by dividing the euclidean distance between them by a constant velocity. Because the quadcopter starts at rest, I noticed the planner commanded large initial accelerations; this large acceleration, compounded with the initial covariance not being zero, caused more drift throughout the fight. I amended this issue by calculating the time to arrive at the initial waypoint with a constant

acceleration, not an initial velocity:

$$t_1 = \sqrt{\frac{2 * dist}{a}} \tag{1}$$

B. Loss of Visual Features for Localization

The VIO algorithm is dependent upon features visible in the camera frame. For this problem, all features greater than 5 meters from the quadcopter are not visible. I encountered issues on the window map as shown in Fig. 2a, where there is a stretch of time with no features visible for VIO. To amend this issue I drove the quadcopter to fly low to the ground (where features are visible because the quadcopter is tilted forward) when there are no obstacles directly ahead. This is achieved by lowering the height of the occupancy grind in stretches of the map with no obstacles within 5m along a heading of 0 degrees (quad flies with fixed yaw). This solution solved the issue of dropped features and was robust across a variety of maps (both provided and self-developed). If the modified occupancy map does not return a path with A*, the height restriction is iteratively reduced until a viable trajectory is returned.



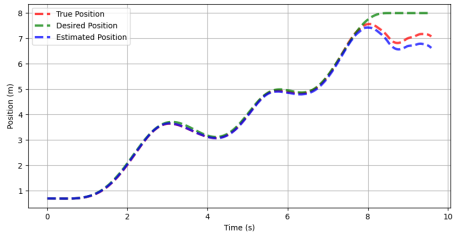
(a) Standard A* (b) A* with flying low fix

Fig. 2: Impact of flying low on quadcopter performance on the ”window” map.

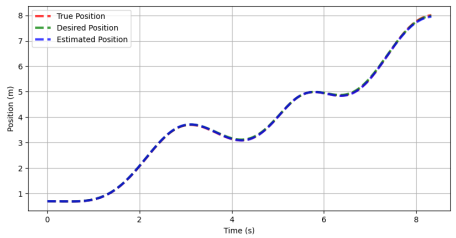
For a future approach, I would consider implementing a yaw controller instead of having the quadcopter fly at a fixed heading. This would enable the planner to control the yaw of the quadcopter to point the cameras in the direction of the most feature rich areas.

C. Controller Gains and Drift

Another major development from projects 1 and 2 to project 3 is controller gains. Even with the above improvements to my software stack, I found that the quadcopter was encountering drift on large maps (such as switchback). To investigate, I developed my own large map to test new ideas on how to fix the issue. Thinking conceptually, I realized that my controller was still tuned for my quadcopter in 1.3. The controller in project 1 could be tuned very tight because the exact position and orientation of the quadcopter was known. In project 3, I found that these gains (especially the inner loop attitude gains) were too aggressive and causing drift error in localization. I spent time reducing the attitude gains on the quadcopter, ultimately settling on values that are smaller by a factor of 3. The result of reducing these gains is shown in Fig. 3.



(a) Project 1 controller gains



(b) Improved project 2 controller gains

Fig. 3: Improvement on estimation drift of quadcopter by reducing controller gains.

III. Performance Results

The final performance of the quadcopter on the auto-grader is given below.

Map	Planning (s)	Flight (s)	Score
maze	2.25	8.14	13.00
over_under	7.71	14.26	13.00
window	13.20	11.71	10.06
stairwell	16.11	15.21	13.00
switchback	17.89	24.15	11.62
slalom	12.67	19.56	13.00

Fig. 4: Test Results

IV. Conclusions and Future Work

I am curious to investigate how well this simulation approach would translate to a real quadcopter. From what we learned from project 1.4 I would expect the quadcopter gains need to be lower to account for delays in the real-world system. Further the real world may not be as feature rich as the simulation. Features will not be evenly spaced on objects, instead likely clustered on the corners and edges.

REFERENCES

[1] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

[2] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

[3] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972.