MEAM6200 Project 3

Enlin Gu

I. Project 3 overview

In Project 3, my main task is to track the trajectory with the State Estimator. The main steps include

1) **World Trajectory Generation:**
   - Generate the desired trajectory with `world_traj.py` using helper functions like `graph_search.py`.
2) **Measure System Setup:**
   - Adjust the import statements in and `vio_utils.py`.
3) **Integrate Estimation and Control:**
   - Refine the SE3 controller and trajectory planner as needed.
   - Ensure that the state estimator integrates with the planner and controller.
4) **Tuning and Testing:**
   - Use `sandbox.py` for analysis and performance tuning.
   - Test the system with provided maps using `util/test.py`.

II. Project 3 Changes

A. Change 1: Spline Implementation

In Project 1, I use line segments. However, in this lab, I have found that line segments can not achieve the goal in this lab.

As can be seen from the figure 1, the bias is increasing very fast with time. This is because there exists a sudden change in the midpoints so there will be a large error, which can bring in a very large error to the measure system. When this error increases with time, the robot will end up with a "goal" that is very far away from the initial goal.

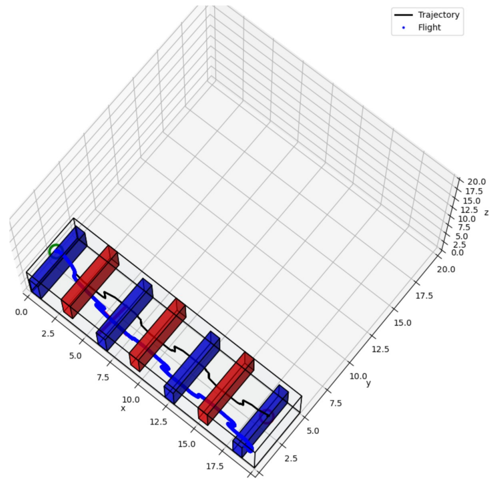

Fig. 1. World Trajectory Using Line Segments

After using a spline, the error will not be as big as what is shown in this figure because the drone position changes much more smoothly. In this lab, I uses a minimum jerk spline for my robot.

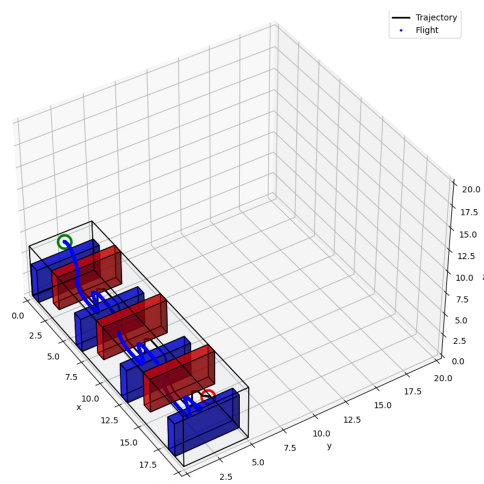

Fig. 2. Trajectory after using spline

This line still has some errors (and will be solved later in another change), but it is much better than before.

B. Change 2: Spline midpoints

In Project 3, I encountered a problem that, my robot would collide with the obstacles even if the velocity is very slow.

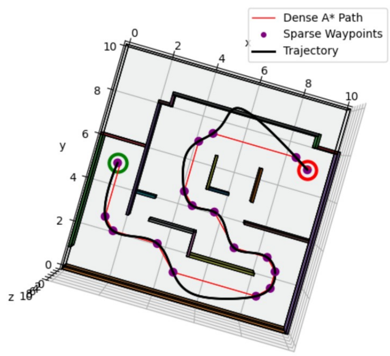

Fig. 3. Spline Colliding with Obsatcle

As can be seen in the figure 3, the issue I encountered is caused by the spline: the spline will collide with the environment because it is trying to draw a curve between two dots far away from each other. The result of the Ramer-Douglas-Peucker Algorithm is too sparse.

To solve this problem, I use a technique that: I will insert a middle point if two waypoints are further than 2 meters away, and repeat it until every two nearby waypoints are within 2 meters. By this algorithm, I successfully solved this problem.

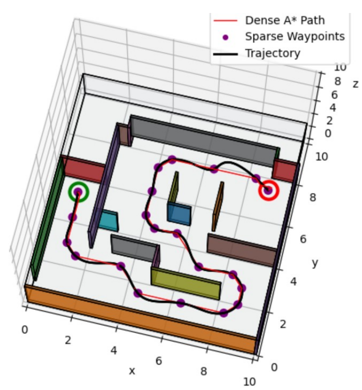

Fig. 4. Spline Successfully Avoid Colliding with Obsatcle

C. Change 3: Covariance matrix of the camera system

As is mentioned in **Change 1**, the end of the trajectory still has an error that can not be ignored. Therefore, I should modify something in the measurement pipeline to decrease the cumulative error.

In this case, I scale the covariance matrix Q_t in VI, so that the error can be smaller. If Q_t is big, there is increasing uncertainty in the model's prediction of the system's behavior. Therefore, there will be large errors with time. In this case, reducing Q_t will decrease the error a little bit.

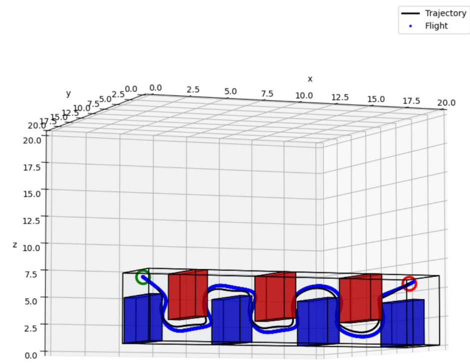

Fig. 5. Small Error Trajectory After Changing Covariance

After changing Q_t, my robot successfully reached the goal within the error tolerance.

D. Minor Changes

Other minor changes include:

- Increase velocity: To make my drone complete the flight tasks as fast as possible, I made it fly faster by tuning the speed to 2.92 m/s.
- Tune my PID controller: I fine-tuned my PID controller again to decrease the error (make my drone follow the trajectory better) and decrease overshoot when arriving at the goal point.
- Increase margin: In this project, because the robot relies on IMU and camera measurements, the system's error will increase with time. This means that we'll need a larger margin to prevent the robot from colliding with obstacles.
- Release clip: In this project, we will have bigger $\ddot{r}$ because we have a relatively bigger error compared with Project 1. In Project 1, I have a safety clip to make sure that the robot's motion will not be too aggressive (so that it will not crash). In this project, I have tuned my PID parameters so that the robot's motion is reliable. However, the clip will make the error of the robot bigger and it can not follow the trajectory very well, so I released this clip.

III. Extra Credit Changes

A. Change 1: Flight Time and Motion Continuity when Re-planning

It took me a long time to find out this issue, so I'd like to discuss it here. It seems that the update function in the local map policy is different from what we did in the first function: At one re-planning point, the flight time is not zero. In this case, if I still use the same update function as in Project 3, the motion will just go crazy.

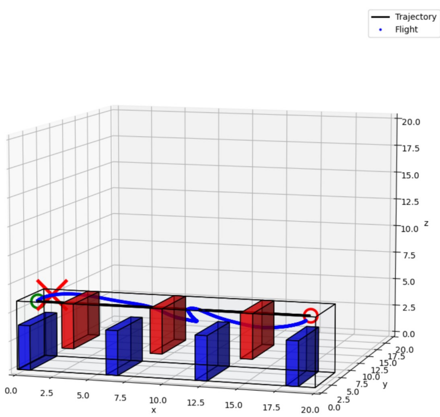

Fig. 6. Crazy Flight with Wrong Time Assigned.

Besides, it is also important to keep the motion continuity between splines: That is, the velocity and acceleration should be recorded. As the system is based on the measurement system to run, a sudden change will bring in errors to the system.

The successful flight looks like the figure below:

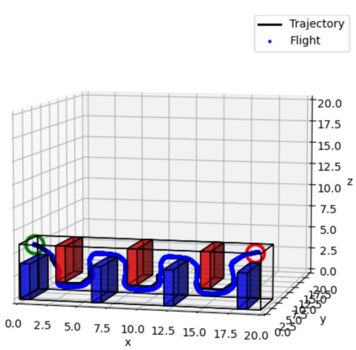

Fig. 7. Successful Flight

B. Minor Changes

Other minor changes include:

- Check the goal of the local map: Sometimes the goal on the local map is within the obstacle, thus the graph search will result in no return. In this case, I use a function to move the goal alone the start-goal line out of the obstacle.
- Decrease speed, so that the robot can follow the trajectory better.

IV. Acknowledgements

I would like to express my deepest gratitude to TA Shiming Liang for assisting me with debugging the spline and guiding me using his method of ”identifying the problem, associating the relevant variable, and understanding how it will change.” His approach made a significant impact on my learning, and his mentorship has been invaluable. He is truly a great teacher.

I would like to thank my classmate Binglong Bao for his dedication in helping me with my code. He generously spent a significant amount of time assisting me in debugging and improving it, providing invaluable support throughout the process.

I would also like to thank my classmates Shuchen Ye and Xiangyu Han for their valuable advice on the spline implementation. Their insights helped me avoid many simple errors, greatly contributing to the success of my work.