Implementation of Planner and Controller

MEAM 620 Project 1 Phase 4

Group 6 : Achin Jain, Yousi Oquendo, Max Gilbert, Eric Young

I. Introduction

In this project, we implemented our control and planning algorithms on a KMel Nano+ quadrotor. Each member had developed their own controller, path-planning algorithms, and trajectory generators in phases 1 through 3 in simulations. The ultimate goal of the final phase and the project as a whole was to test these on the actual hardware for some unknown trajectories.

The project in total consisted of four parts. In phase 1, each team member individually developed a PD controller and tuned gains according to the performance of a simulation of the Nano+ quadrotor. Phase 2 involved the implementation of two path-planning algorithms, Dijkstra and A*. Phase 3 required each member to individually implement a trajectory generator that allowed the quadrotor to follow the path assigned by the A* algorithm. In phase 4, components of each phase were combined and optimized to most effectively fly the physical Nano+ quadrotor in the flight lab.

This report is the summary of the phase 4 group project. In Section II and III, we give a description of our controller and trajectory generator used during the actual implementation, respectively. Section IV contains links to the videos of each experiment, followed by plots to demonstrate the algorithm performance. Here, we also discuss our results, describe factors that could be improved for better performance as well as factors that led to difficulties in the lab.

II. Controller Design

Since the on-board controller took care of the quadrotor's attitude, we only need to calculate the desired angles ϕ_{des} and θ_{des}, and then implement a controller for the position of the quadrotor. From the Vicon and higher-level command, our PD controller is given the following inputs: actual position x, desired position x_{des}, actual velocity v, desired velocity v_{des}, desired acceleration a_{des}, actual yaw angle ψ, desired yaw angle ψ_{des}. For our implementation on the real quadrotor, we use the proportional gain $k_p = [11, 11, 11]^T$, and the derivative gain $k_d = [6, 6, 6]^T$.

The controller first calculates the commanded acceleration a_c using the control equation

$$\ddot{e} + k_d \dot{e} + k_p e = 0, \tag{1}$$

where the error e is defined as $e = x_{des} - x$. The commanded acceleration is the given by

$$a_c = a_{des} + k_d(v_{des} - v) + k_p(x_{des} - x). \tag{2}$$

The desired force output of the motors F is then calculated as follows:

$$F = \frac{m(g + a_{c,z})}{1000g}, \tag{3}$$

where $a_{c,z}$ is the z-component of the acceleration, and a factor of 1000 is used to convert the force in Newton to gram. This is true close to the point of linearization where roll ϕ and pitch θ are close to 0.

Finally, the controller calculates the desired roll ϕ_{des} and desired pitch θ_{des}. From Newton's law of motion we know

$$a_c = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \frac{\mathbf{F}}{m} \end{bmatrix} \tag{4}$$

$$R = \begin{bmatrix} C\theta C\psi & C\psi S\phi S\theta - C\phi S\psi & S\phi S\psi + C\phi C\psi S\theta \\ C\theta S\psi & C\phi C\psi + S\phi S\theta S\psi & C\phi S\theta S\psi - C\psi S\phi \\ -S\theta & C\theta S\phi & C\phi C\theta \end{bmatrix}, \tag{5}$$

where C and S represent *cos* and *sin* functions, respectively. Eq. (4) relates x-component of acceleration $a_{c,x}$ and y-component of acceleration $a_{c,y}$ to ϕ_{des} and θ_{des}:

$$\begin{bmatrix} a_{c,x} \\ a_{c,y} \end{bmatrix} = \begin{bmatrix} g(\theta \cos\psi + \phi \sin\psi) \\ g(\theta \sin\psi - \phi \cos\psi) \end{bmatrix}. \tag{6}$$

It is rearranged to give

$$\begin{bmatrix} \phi_{des} \\ \theta_{des} \end{bmatrix} = \begin{bmatrix} \frac{1}{g}(a_{c,x} \sin\psi - a_{c,y} \cos\psi) \\ \frac{1}{g}(a_{c,y} \sin\psi + a_{c,x} \cos\psi) \end{bmatrix}. \tag{7}$$

The final output of our controller includes $F, \phi_{des}, \theta_{des}$, and ψ_{des}.

III. Navigation Through Way Points

From the given set of way points *path*, we first filter the points to include only points at which a directional change is observed using the method below:

```
$\epsilon = 0.01$
$WayPts = 1$
for $i = 2 : NoWayPts - 1$ do
    $x_i \leftarrow path(i,1)$
    $y_i \leftarrow path(i,2)$
    $z_i \leftarrow path(i,3)$
    if any($|2x_i - x_{i-1} - x_{i+1}|, |2y_i - y_{i-1} - y_{i+1}|, |2z_i - z_{i-1} - z_{i+1}| > \epsilon$) then
        $WayPts \leftarrow [WayPts,\ i]$
    end if
end for
$WayPts \leftarrow [WayPts,\ NoWayPts]$
```

We then obtained smooth trajectories in between the filtered way points $WayPts$. In order to better assess the performance of our controller and better understand the wide assortment of trajectory generators available, we tested two different methods of calculating the trajectories. The first used a quintic polynomial with zero velocity and zero acceleration assigned at each way point. The second used a cubic polynomial which maintains velocity and acceleration continuity at each way point, and has zero velocity at start and stop.

Decoupling of the Euler-Lagrange equations allows us to fit the polynomials to x, y, and z dimensions independently. We have given the procedure for each method in the x dimension, but the set of equations (8)-(18) also hold for y and z on the same time scale. For the quintic trajectories, applying the Euler-Lagrange equation to

$$x^*(t) = \arg\min_{x(t)} \int_0^T (\dddot{x})^2 dt \tag{8}$$

yields a 5^{th} order polynomial $c_0+c_1t+c_2t^2+c_3t^3+c_4t^4+c_5t^5$, whose coefficients can be calculated by solving the matrix equation

$$\begin{bmatrix} x_i \\ x_f \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}, \tag{9}$$

with boundary conditions

$$x(0) = x_i,\ x(T) = x_f, \tag{10}$$
$$\dot{x}(0) = 0,\ \dot{x}(T) = 0, \tag{11}$$
$$\ddot{x}(0) = 0,\ \ddot{x}(T) = 0. \tag{12}$$

Different splines are designed for each segment in $WayPts$ using the corresponding x_i and x_f, thus giving a piecewise trajectory. Time T is scaled according to the value of $|x_i - x_f|$.

Derivation of the cubic trajectories follows a slightly different procedure. In this case, minimizing acceleration instead of jerk

$$x^*(t) = \arg\min_{x(t)} \int_0^T (\ddot{x})^2 dt \tag{13}$$

yields a 3^{rd} order polynomial. Let us define the polynomial describing the segment between way points $(k-1)$ and k as $x_k(t) = c_{0,k} + c_{1,k}t + c_{2,k}t^2 + c_{3,k}t^3$. However, this time the velocities and accelerations at each way point are unknown, thus the segments of the trajectory cannot be decoupled. So, the boundary conditions become

$$x_1(0) = x_0,\ x_n(T_n) = x_f, \tag{14}$$
$$\dot{x}_1(0) = 0,\ \dot{x}_n(T_n) = 0, \tag{15}$$

$$x_k(T_k) = x_k = x_{k+1}(T_k), \tag{16}$$
$$\dot{x}_k(T_k) = \dot{x}_{k+1}(T_k), \tag{17}$$
$$\ddot{x}_k(T_k) = \ddot{x}_{k+1}(T_k), \tag{18}$$
$$\forall k = 1, ..., n-2.$$

IV. Experimental Results and Discussion

In total we performed 6 experiments. The links to the videos are embedded in the headings.

1) Hover: This is shown in Fig. 1. Beyond 7 s, the quadrotor is in hovering at $z = 1.9$ m (approx). Actual x, y and z are slightly different from the desired values. This suggests a better tuning of gains is required.

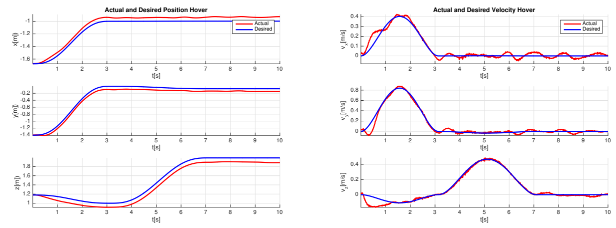

Fig. 1: Left: Desired vs Actual Position for Hover. Right: Desired vs Actual Velocity for Hover.

2) Go2WayPt: Fig. 2 shows this trajectory which is a straight line from the current position to the final position. So we use just one quintic spline between 2 way points. In the velocity graph, we can see that the v_x, v_y and v_z are 0 at start and end.

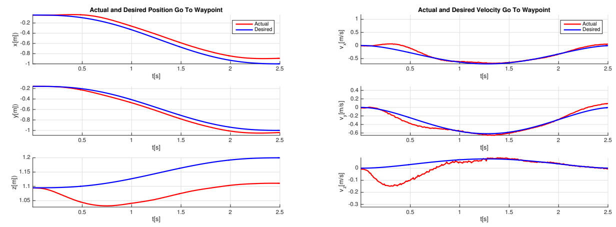

Fig. 2: Left: Desired vs Actual Position for Go2WayPt. Right: Desired vs Actual Velocity for Go2WayPt.

3) Trajectory 1: This trajectory is also a straight line. Since our current position before the start of the trajectory was not in the same line, we first follow a straight line to go to the start point, and then another to go to the end point. So, in total, we have 2 quintic splines connecting 3 ways points, which is verified from the velocity graph (Fig. 3) since v_x, v_y and v_z are all 0 at 0 s, 3 s and then 6 s. The time T for both the splines was fixed at 3s. 3-D trajectory in Fig. 9 also shows 2 straight line paths.

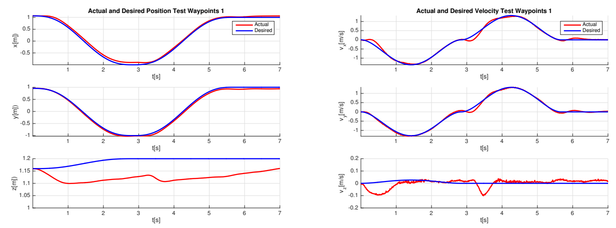

Fig. 3: Left: Desired vs Actual Position for Trajectory 1. Right: Desired vs Actual Velocity for Trajectory 1.

4) Trajectory 2: This trajectory is a collection of many straight line paths and sharp corners. So, we again tested quintic spline to keep the velocities at the way points equal to 0. During this test, we lost Vicon tracking momentarily, which is visible at 28 s and 43 s in Fig. 4. We also observe significant noise in velocity measurements from the Vicon, but the position is tracked fairly well.

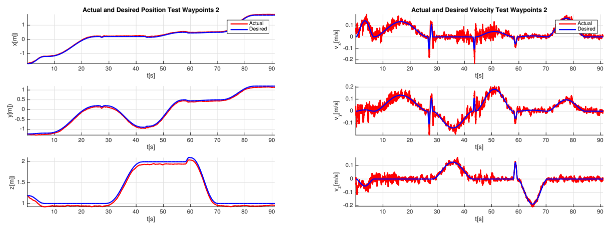

Fig. 4: Left: Desired vs Actual Position for Trajectory 2. Right: Desired vs Actual Velocity for Trajectory 2.

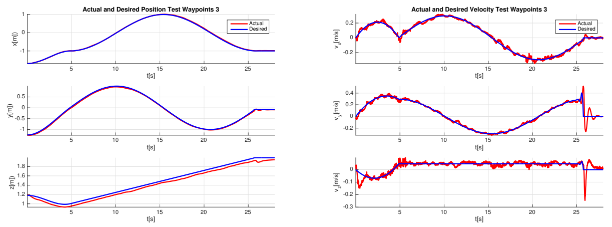

Fig. 5: Left: Desired vs Actual Position for Trajectory 3 (Fast). Right: Desired vs Actual Velocity for Trajectory 3 (Fast).

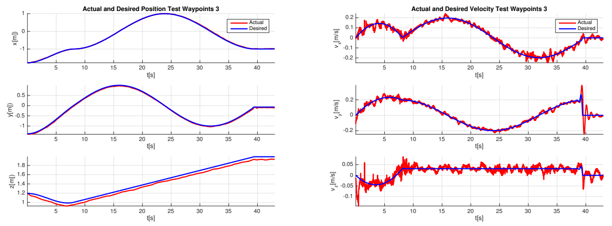

Fig. 6: Left: Desired vs Actual Position for Trajectory 3 (Slow). Right: Desired vs Actual Velocity for Trajectory 3 (Slow).

5) Trajectory 3: In this case, the trajectory is a helix. So, it makes more sense to have nonzero velocity and acceleration at intermediate way points to have a smooth trajectory, which makes the cubic spline described by (13)-(18) a natural choice. Using the filtering method described in Section III, we could not track the trajectory initially. It turned out that ϵ was too high for the given way points and the complete set of points reduced to just 2 points. Reducing ϵ to 0.001 fixed the issue. We experimented by varying average speed, hence the time T for the cubic polynomial. In Fig. 5, the trajectory is completed in 26 s and in Fig. 6 in 41 s. Again, the velocity measurements from Vicon are very noisy.
6) Trajectory 4: It is the most complex trajectory with a zig-zag path which required us to iterate over the average speed because at higher speeds, we reached maximum actuation limits. In this case, we compared both cubic and quintic splines. Because the trajectory has sharp turns, cubic spline (Fig. 7) smoothens the corners while with quintic spline (Fig. 8) we visit each way point closely with 0 velocity and acceleration. Loss in Vicon tracking is again evident at 27 s in Fig. 8.

Tab. I lists the total and average errors in position in the x, y, and z directions, and Tab. II lists the total and average errors in velocity in the x, y, and z directions. Total errors were found by subtracting the desired from the actual x, y, and z positions and velocities and taking the absolute sum of all differences. The average errors were found by taking the average of all of the differences in positions and velocities. The largest position errors in each trial tended to be in the z direction, which is due to the safety offset in our code and the inherent steady state error due to a lack of integral control. This systematic error could have been minimized by using a PID controller rather than a PD controller. The steady state error could have, in an ideal world, also been minimized by increasing the proportional gain term, but we did not have enough time to optimize our gains. The gains used in the

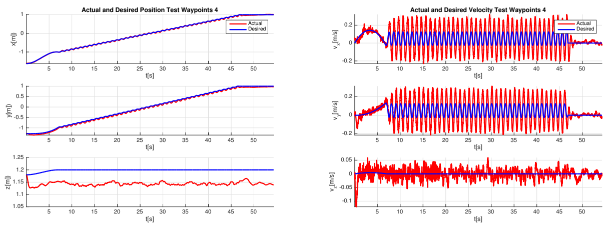

Fig. 7: Left: Desired vs Actual Position for Trajectory 4 (Cubic). Right: Desired vs Actual Velocity for Trajectory 4 (Cubic).

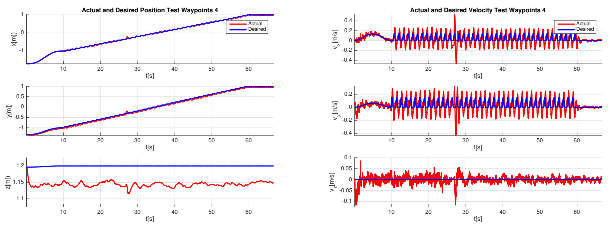

Fig. 8: Left: Desired vs Actual Position for Trajectory 4 (Quintic). Right: Desired vs Actual Velocity for Trajectory 4 (Quintic).

TABLE I: Position Errors.

Path	Type	X pos Tot error, m (Avg error, m)	Y pos Tot error, m (Avg error, m)	Z pos Tot error, m (Avg error, m)
Hover	Q	240.3133 (0.0649)	256.0162 (0.0691)	325.5915 (0.0879)
OneWP	Q	29.7784 (0.0897)	18.1723 (0.0547)	28.4612 (0.0857)
WayPts1	Q	189.0451 (0.0570)	163.9992 (0.0494)	199.4702 (0.0199)
WayPts2	Q	346.6091 (0.0381)	502.8545 (0.0553)	530.2195 (0.0583)
WayPts3	C (slow)	45.2626 (0.0136)	97.8658 (0.0293)	175.8017 (0.0527)
WayPts3	C (fast)	59.9977 (0.0110)	164.0733 (0.0299)	294.7929 (0.0538)
WayPts4	C	98.7113 (0.0182)	208.8260 (0.0384)	299.7928 (0.0552)
WayPts4	Q	99.4459 (0.0149)	295.4864 (0.0443)	373.2500 (0.0560)

TABLE II: Velocity Errors.

Path	Type	X lin vel Tot error, m/s (Avg error, m/s)	Y lin vel Tot error, m/s (Avg error, m/s)	Z lin vel Tot error, m/s (Avg error, m/s)
Hover	Q	64.2575 (0.0173)	58.0833 (0.0157)	41.6170 (0.0112)
OneWP	Q	47.9674 (0.1445)	39.0884 (0.0548)	18.1784 (0.0548)
WayPts1	Q	66.1297 (0.0199)	60.0486 (0.0181)	39.6125 (0.0119)
WayPts2	Q	126.4460 (0.0139)	153.3425 (0.0169)	91.5661 (0.0101)
WayPts3	C (slow)	43.0852 (0.0129)	62.2687 (0.0187)	46.0751 (0.0138)
WayPts3	C (fast)	74.0170 (0.0135)	84.4557 (0.0154)	58.9991 (0.0108)
WayPts4	C	433.1966 (0.0797)	411.2992 (0.0757)	73.7797 (0.0136)
WayPts4	Q	460.9477 (0.0691)	464.6398 (0.0697)	68.7832 (0.0103)

TABLE III: Controller Gains.

Gain Type	X	Y	Z
Kp (Quadrotor)	11	11	11
Kd (Quadrotor)	6	6	6
Kp (Simulation)	22	22	200
Kd (Simulation)	10	10	50

experiments are listed in Tab. III. The table also shows the corresponding values for simulations in Phase 1 of the project which were found to be too high for real experiments.

While total position errors varied dramatically due to the length of each trajectory, the average position errors per path point were between 1 and 9 cm in each trajectory (Tab. I). In nearly every trajectory, the z-position exhibited the largest average errors. Total velocity errors similarly varied dramatically by length of trajectory, but average velocity errors again stayed between 1 and 9 cm/s. In Trajectory 3, the slow cubic polynomial resulted in a smaller total error as expected, but the both algorithms exhibited similar average errors. The trajectory showed similar trends in velocity (Tab. II). In Trajectory 4, the cubic polynomial with non-zero waypoint velocities had smaller total errors and smaller average errors in all directions with the exception of the x direction. The velocities showed similar smaller average errors for the quintic polynomial with zero waypoint velocities and smaller total errors for the cubic polynomials with zero velocities at waypoints.

Certain errors in the experiments were due to factors out of the control of our group, such as the use of a linearized controller and the systems used for tracking and control. Additionally, in our first lab session our group used a variable (time0) which is a global variable in the system. It took us around 1 hr to figure out the issue, as a result we couldn't really achieve anything significant at the end of our first session. The time it took to discover could have been used to further tune the gains and improve the trajectory generator functions. During the last (makeup) session, we also faced troubles connecting to Vicon and sometimes we lost tracking completely in the middle of experiments. Morever, there was

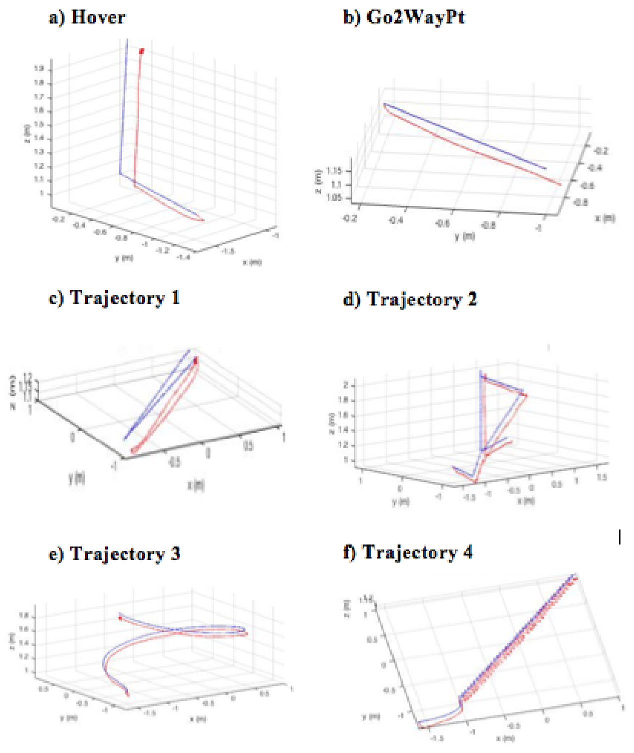

Fig. 9: All trajectories in 3-D.

no weighing scale to measure the weight of the quadrotor in the last session when we generated most of our results. The error in z direction is also attributed to this factor.

V. Conclusion

The performance of the quadrotor was somewhat limited due to the use of a linear controller, especially with the more aggressive trajectories that resulted in larger changes in direction. Due to the small angle approximation used to linearize the controller, trajectories with many turns and larger changes in direction often had larger errors in velocity tracking, as well as larger position overshoots at corners. The performance was also limited due to the lack of an integral controller, which would have decreased steady state error and allowed for closer tracking in the z direction. External factors such as unknown global variables set by the teaching team, difficulties with Vicon tracking, and limited lab time also limited optimal performance.