TRAJECTORY GENERATION, COMPARISON AND CONTROL FOR QUADCOPTERS

Praveen Jawaharlal Ayyanathan

***Abstract*— Quadcopters are now used extensively to reach difficult destinations like forest fires or flooded regions. In this project minimum jerk (a 5^{th} degree polynomial trajectory with 3^{rd} derivative cost minimization) and minimum snap (a 7^{th} degree polynomial trajectory with 4^{th} derivative cost minimization) trajectories for a quad-copter are developed that will allow for its smooth motion. The minimum jerk and minimum snap trajectories are also compared to understand polynomial trajectories. Then time allocation for each node is performed that will minimize the total time of flight between two nodes. A PD controller is also developed to track the desired polynomial trajectory.**

I. INTRODUCTION

Trajectory planning and control is a crucial step in realizing autonomous quadcopters that are now increasingly used in surveillance, reconnaissance, industry inspection, search and rescue missions. A quadcopter is an underactuated system with four control inputs to handle six degrees of freedom. The translational motion of a quadcopter is coupled with its rotational motion making its dynamics complex. Since the computational power of the on-board controller is limited, it is important to generate trajectory and control profile efficiently. A delayed response from the quadcopter might result in its crash. Therefore it is necessary to perform trajectory planning quickly. One way to realize quicker trajectory planning is to make use of the differential flatness property. A system is said to have differential flatness property when the state and input of the dynamical system can be mapped into a set of flat output variables and their derivatives. Mellinger and Kumar [1] showed that the quadrotor dynamics with the four inputs ($[x, y, z, \psi]$) is differentially flat (where $\boldsymbol{r} = [x, y, z]$ are the coordinates of the center of mass in the inertial frame and ψ is the heading angle). Mellinger and Kumar's trajectory planning technique enables fast algebraic calculation of the trajectory and since we can enforce the differentiability of the polynomial in the cost function, a smooth continuous trajectory is also possible.

II. POLYNOMIAL TRAJECTORY GENERATION

The optimal trajectory profile $x^*(t)$ is one that minimizes

$$x^*(t) = \arg\min_x \int_0^{t_f} L(t, x, x^{(2)}, .., x^{(N)})dt, \quad (1)$$

where t_f denotes the final time and N denotes the number of times the trajectory is differentiable. For minimum jerk trajectory $L = x^{(3)} \times x^{(3)}$ and for minimum snap trajectory $L = x^{(4)} \times x^{(4)}$. The Euler Lagrange equation upto N^{th} degree can be written as

$$\frac{\partial L}{\partial x} + \sum_{k=1}^{N} (-1)^k \frac{d^k}{dt^k} \left(\frac{\partial L}{\partial x^{(k)}} \right) = 0. \quad (2)$$

By substituting the Lagrangian in Eq. (2) we will get a Polynomial segment of degree $2N - 1$. For minimum jerk the polynomial is of the form $c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$ and for minimum snap the polynomial is of the form $c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 + c_6 t^6 + c_7 t^7$. If $\boldsymbol{p} = [c_0 \ c_1 \ c_2 \ .... \ c_{2N-1}]^\top$ represents a column vector of the coefficients of the polynomial, the optimal trajectory profile x^* can be written as

$$x^*(t) = \arg\min_x \int_0^{t_f} \boldsymbol{p}^\top \boldsymbol{Q} \boldsymbol{p}, \quad (3)$$

where $\boldsymbol{Q}$ is a cost matrix. This form of cost can be solved using the techniques of a standard Quadratic Programming (QP).

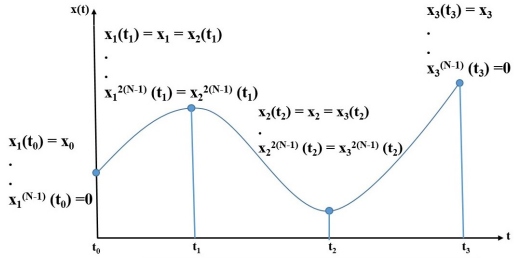

Fig. 1: General form of constraints for polynomial trajectory optimization.

III. Constraints

To find the extremal polynomial trajectory it is required to find the elements of the column vector $\boldsymbol{p}$. Therefore for a single segment we need to have $2N$ constraints as there are $2N$ coefficients. Also, constraints are enforced on each node of the polynomial when multiple nodes are present. If the number of segments is denoted by m, then we need to have $2Nm$ constraints in total. Position data of each node is generally known apriori and we have $2m$ position constraints. Further it is necessary to include continuity constraints at the intersection of two segments so as to have a smooth trajectory. We can also enforce constraints up to $(N-1)^{th}$ order at the start and end nodes to make the start and end of the trajectory according to the mission statement. All the constraints are summarized in Figure 1.

IV. Time Allocation

The time at each node can be fixed manually but it may not give the optimal solution. The time associated with each segment can be made to vary to improve the overall solution with respect to a cost function. We can optimize the total time by adding it to the cost function as

$$\mathrm{J}_{\mathrm{tot}} = \boldsymbol{p}^T Q \boldsymbol{p} + K_\tau \sum_{i=1}^{m} \tau_i, \tag{4}$$

where K_τ is a user-specified penalty on time. The first term in this cost function is the original cost function for polynomial optimization. In this project K_τ is taken a 1. One of the advantage of using time allocation is that overshooting of trajectories between two nodes can be avoided. Another advantage of not selecting the total time arbitrarily is that by penalizing the time of particular node points, it is possible to automatically slow down the quadcopter while navigating tightly spaced obstacles.

V. Controller Description

A 3D-trajectory PD controller [2] is developed to realize the minimum snap trajectory that is developed. The simulated 3D-trajectory controller is applicable for modest accelerations as near-hover assumptions are considered. Let $\boldsymbol{r_T} = [r_{1,T}, r_{2,T}, r_{3,T}]^\top, \boldsymbol{\dot{r}_T} = [\dot{r}_{1,T}, \dot{r}_{2,T}, \dot{r}_{3,T}]$, and $\boldsymbol{\ddot{r}_T} = [\ddot{r}_{1,T}, \ddot{r}_{2,T}, \ddot{r}_{3,T}]$ denote the target position, velocity and acceleration respectively. The error in position and velocity is written as

$$\begin{aligned} e_i &= (r_{i,T} - r_i), \ i = 1, 2, 3, \\ e_i &= (\dot{r}_{i,T} - \dot{r}_i). \end{aligned} \tag{5}$$

The desired acceleration can be obtained using

$$\begin{aligned} \ddot{r}_{1,\text{ des}} &= \ddot{r}_{1,T} + k_{d,1}\left(\dot{r}_{1,T} - \dot{r}_1\right) \\ &\quad + k_{p,1}\left(r_{1,T} - r_1\right), \\ \ddot{r}_{2,\text{ des}} &= \ddot{r}_{2,T} + k_{d,2}\left(\dot{r}_{2,T} - \dot{r}_2\right) \\ &\quad + k_{p,2}\left(r_{2,T} - r_2\right), \\ \ddot{r}_{3,\text{ des}} &= \ddot{r}_{3,T} + k_{d,3}\left(\dot{r}_{3,T} - \dot{r}_3\right) \\ &\quad + k_{p,3}\left(r_{3,T} - r_3\right), \end{aligned} \tag{6}$$

where $k_{d,i}$ and $k_{p,i}$ are the tuning parameters. The desired roll, pitch and yaw angles can be obtained using

$$\begin{aligned} \phi_{\mathrm{d}} &= \frac{1}{g}\left(\ddot{r}_{1,\mathrm{des}} \sin\psi_T - \ddot{r}_{2,\mathrm{des}} \cos\psi_T\right), \\ \theta_{\mathrm{d}} &= \frac{1}{g}\left(\ddot{r}_{1,\mathrm{des}} \cos\psi_T + \ddot{r}_{2,\mathrm{des}} \sin\psi_T\right), \\ \psi_{\mathrm{d}} &= \psi_T(t), \end{aligned} \tag{7}$$

where ψ_T is the target yaw profile. If u_1 and $\boldsymbol{u_2}$ denote the force and moment controls, then

$$\begin{aligned} u_1 &= mg + m\ddot{r}_{3,\mathrm{des}}, \\ \boldsymbol{u}_2 &= \begin{bmatrix} k_{p,\phi}\left(\phi_{\mathrm{d}} - \phi\right) + k_{d,\phi}\left(p_{\mathrm{d}} - p\right) \\ k_{p,\theta}\left(\theta_{\mathrm{d}} - \theta\right) + k_{d,\theta}\left(q_{\mathrm{d}} - q\right) \\ k_{p,\psi}\left(\psi_{\mathrm{d}} - \psi\right) + k_{d,\psi}\left(r_{\mathrm{d}} - r\right) \end{bmatrix}. \end{aligned} \tag{8}$$

VI. NUMERICAL RESULTS

A. Comparing Minimum Jerk and Minimum Snap Trajectories

The minimum jerk and minimum snap trajectories are treated as QP and solved using MATLAB's *quadprog*. A three segment trajectory (therefore four node points) is considered with the positions and heading of the node points taken as

$$\begin{aligned} x_{nodes} &= [0, 2, 3, 1]^\top \text{units}, \\ y_{nodes} &= [0, 2, 3, 1]^\top \text{units}, \\ z_{nodes} &= [0, 2, 3, 1]^\top \text{units}, \\ heading_{nodes} &= [0, 2, 3, 1]^\top \text{units}. \end{aligned} \tag{9}$$

Initially time allocation is not used (this is done so that the minimum jerk and minimum snap trajectories can be compared and analyzed) and the time at each node is considered as

$$t_{nodes} = [0, 1, 3, 5]^\top \text{units}. \tag{10}$$

The minimum snap and minimum jerk trajectories are shown in Figure 2. The y, z and angle trajectory profiles look the same as the x-trajectory since the position of the nodes are the same.

Next a combination of sine and cosine like trajectories are simulated and position and time node points are taken as

$$\begin{aligned} x_{nodes} &= [0, 2, 0, -2]^\top \text{units}, \\ y_{nodes} &= [0, 1, 0, -1]^\top \text{units}, \\ z_{nodes} &= [1, 0, -1, 0]^\top \text{units}, \\ t_{nodes} &= [0, 1.5, 4.5, 6]^\top \text{units}. \end{aligned} \tag{11}$$

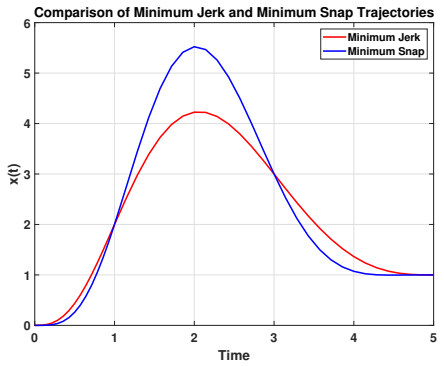

Fig. 2: Minimum snap and minimum jerk trajectories when node points are taken as $[0, 2, 3, 1]^\top$.

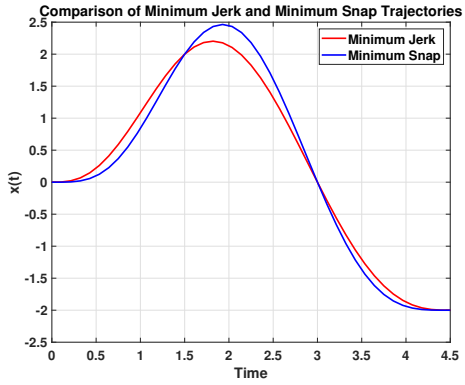

Fig. 3: Minimum snap and minimum jerk trajectories when node points are taken as $[0, 2, 0, -2]^\top$.

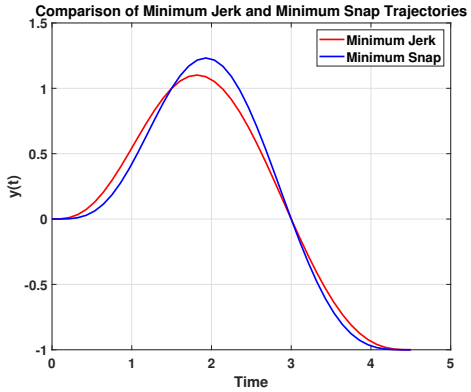

Fig. 4: Minimum snap and minimum jerk trajectories when node points are taken as $[0, 1, 0, -1]^\top$.

The x, y and z trajectory profiles of minimum snap and minimum jerk trajectories are shown in Figure 3, Figure 4 and Figure 5 respectively. The trajectory plots suggest that the minimum snap trajectory always overshoots more than the minimum jerk trajectory after a given node point. This is because the minimum snap continuity constraint is two orders more than the minimum jerk continuity constraint (refer Figure 1). Figure 6 shows the complete minimum snap and minimum jerk 3-D trajectory.

B. Time Allocation and Obstacle Avoidance

From Section (VI-A) it is evident that the minimum snap trajectory overshoots near the node points to maintain a smooth trajectory, and therefore it is important not to select a node point near an obstacle. To include time minimization into the simulation, the formulation shown in Section (IV) is followed. It was found that once time allocation is included in the trajectory optimization formulation, MATLAB's *quadprog* cannot be used. Therefore MATLAB's *fmincon* is used. Two predefined hoops are defined in a 3D space in MATLAB. The coordinates of the hoops are

$$\begin{aligned} \text{hoop}_1 &= [1, 1, 0]^\top \text{units}, \\ \text{hoop}_2 &= [3, 3, 0.5]^\top \text{units}. \end{aligned} \tag{12}$$

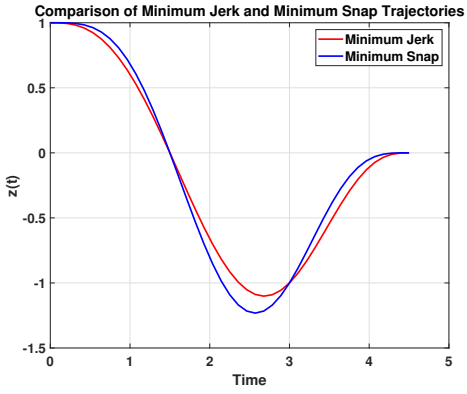

Fig. 5: Minimum snap and minimum jerk trajectories when node points are taken as $[1, 0, -1, 0]^\top$.

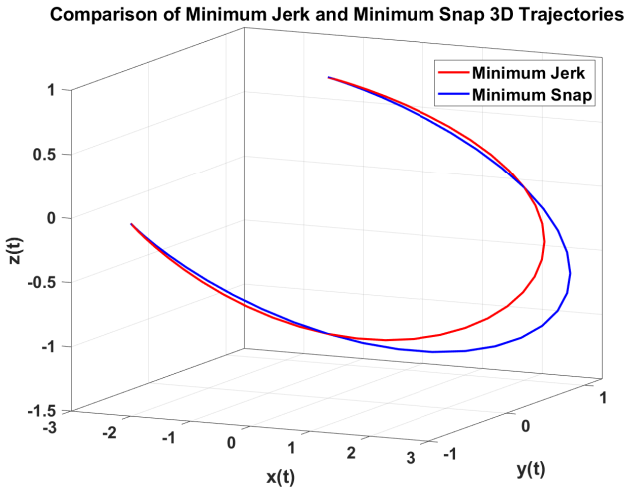

Fig. 6: Minimum snap and minimum jerk 3D trajectories.

The aim is to go through the hoops without touching its boundaries. The x and y node positions can be chosen in the 3D plot using MATLAB GUI. Care should be taken to not take the nodes near the hoop boundaries. The z nodes are taken as $[-0.5; 0.25; 0.75; 0]^\top$. Initial and final time are taken as $t_0 = 0s$ and $t_f = 6s$ respectively. The constraints for time allocation are: 1) the minimum time taken to move from one node to another should not be less than one unit, and 2) the maximum time taken to move from one node to another should not be more than three units.

Using the GUI, x and y nodes are selected as $[0.0426; 1.3905; 1.7822; 3.4758]^\top$ and $[-0.0743; 1.2813; 2.3309; 3.4679]^\top$ respectively. Time allocation at each node found using $fmincon$ is $[0, 1.9557, 3.0000, 5.9999]^\top$. The simulated 3D-trajectory is shown in Figure 7.

C. CONTROLLER IMPLEMENTATION

The PD controller discussed in Section (V) is implemented in MATLAB. The trajectory profile and dynamics were taken from [2] and a controller code file was written. The polynomial generated using the minimum snap approach can be also be used to generate the trajectory. The tuned parameter that were used during the simulation are,

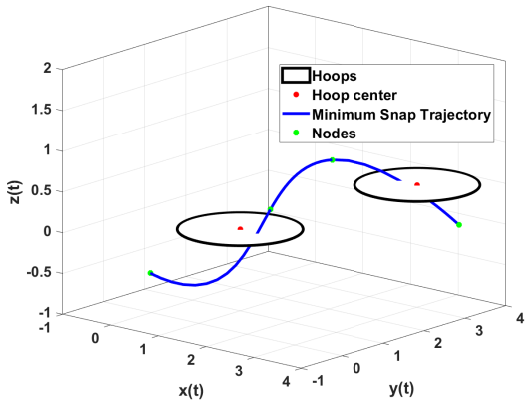

Fig. 7: Minimum snap trajectory passing through the two hoops without colliding.

$$
\begin{aligned}
k_{d,1} = 0.5, k_{d,2} = 0.5, k_{d,3} = 0.5, \\
k_{p,1} = 4.5, k_{p,2} = 4.5, k_{p,3} = 4.5, \\
k_{d,\phi} = 0.5, k_{d,\theta} = 0.5, k_{d,\psi} = 0.5, \\
k_{p,\phi} = 75, k_{p,\theta} = 100, k_{p,\psi} = 20.
\end{aligned} \tag{13}
$$

The inbuilt helix trajectory was simulated and the simulation results are shown in Figure 8, Figure 9 and Figure 10. The controller is able to follow the desired trajectory exactly along the x- and y-axis. But there is a small difference seen in the z-axis between $5-9$ seconds. Several values for the tuning parameters were tried but the z-axis trajectory is associated with a small error (about 0.1m in position).

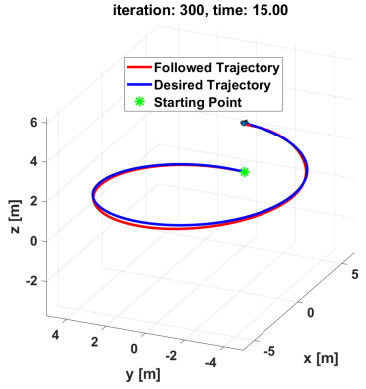

Fig. 8: 3D trajectory obtained by the controller.

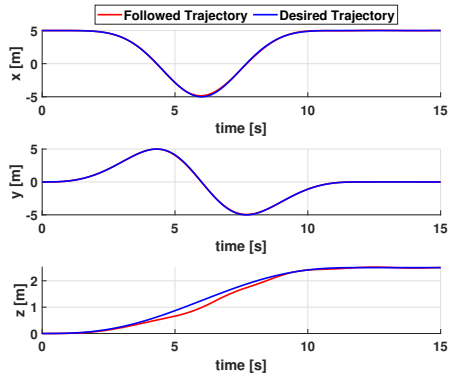

Fig. 9: Position Comparison of the followed and desired trajectory.

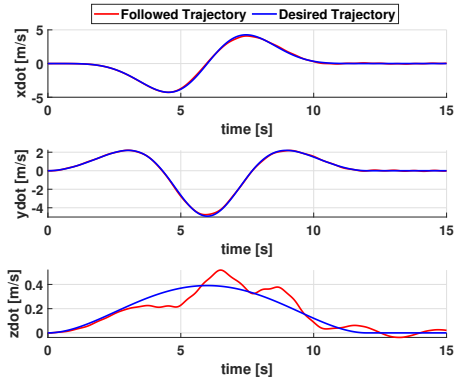

Fig. 10: Velocity Comparison of the followed and desired trajectory.

VII. CONCLUSION

Minimum snap and minimum jerk trajectories are developed and it is seen that the former trajectory has more overshooting near the node points than the latter trajectory. This overshooting should be kept in mind while selecting node points near obstacles. Then time allocation is introduced into the formulation and it was observed that the problem is no longer solvable using MATLAB's *quadprog*. Then a PD controller is developed and it was observed that the x- and y- trajectory exactly followed the predefined trajectory but there are some error in the z- trajectory.

REFERENCES

[1] D. Mellinger, V. Kumar, Minimum snap trajectory generation and control for quadrotors, in: 2011 IEEE international conference on robotics and automation, IEEE, 2011, pp. 2520–2525.

[2] V. Kumar, Robotics: Aerial robotics, in: Coursera, University of Pennsylvania.