Optimal Control for Autonomous Drone Racing

Aleix Paris[1]

***Abstract*— This paper studies the problem of optimal control of a quadrotor to minimize the time it takes it to pass trough several waypoints, that is, to finish a race.**

I. INTRODUCTION

Optimal control problems have been widely studied... The Red Bull Air Race, where airplanes cross gates to end a circuit as fast as possible is an example of a similar problem that has been studied... Nowadays, every team has the role of a 'tactician', which is in charge of...

MENTION PARKER AND AUTONOMOUS DRONE RACING COMPETITIONS...

The paper is structured as follows: Section II presents the formulation of this problem, Section...

II. PROBLEM FORMULATION

Several authors have studied quadrotor dynamics [1]. This section presents the assumptions presumed, the geometry of a quadrotor, its dynamics, the state-space model used, and the mathematical formulation of the problem in hand.

A. Assumptions

The formulation that follows considers the following assumptions:

1) The maximum and minimum altitudes of the quadrotor are similar and thus the air density and gravity are constants.
2) The vehicle is flying in zero-wind conditions.
3) No ground effect is considered.
4) The angular velocities of the rotors are similar, and the rotors' inertia is small.
5) The quadrotor is symmetrical with its four arms aligned with the body x- and y-axes.
6) The propellers are rigid and thus blade flapping does not occur.

B. Quadrotor Geometry and Notation

The quadrotor's absolute linear position is defined in the inertial frame with the vector $\boldsymbol{\xi}$. Similarly, the attitude (angular position of the drone with respect to the inertial frame) is defined with the vector $\boldsymbol{\eta}$. Roll angle ϕ determines the rotation of the vehicle around the x-axis, pitch angle θ defines a rotation around the y-axis, and yaw angle ψ determines the quadrotor's rotation around the z-axis:

$$\boldsymbol{\xi} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{\eta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$$

[1]Graduate Research Assistant at the Aerospace Controls Laboratory, MIT, 77 Massachusetts Ave., Cambridge, MA, USA aleix@mit.edu

The origin of the body frame, indicated with a B, is the center of mass of the quadrotor. In this frame, the vehicle's linear velocities $\boldsymbol{V_B}$ and angular velocities $\boldsymbol{\nu}$ are:

$$\boldsymbol{V}_B = \begin{bmatrix} v_{x,B} \\ v_{y,B} \\ v_{z,B} \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Figure 1 shows the inertial and body frame, as well as the Euler and angular velocity angles defined previously.

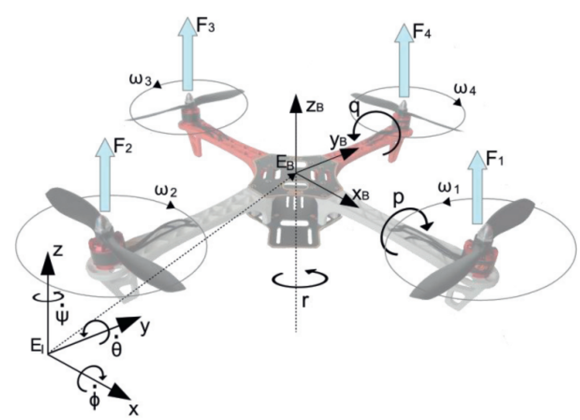

Fig. 1. Inertial and body-fixed frame of the quadrotor, showing the Euler and angular velocity angles [?]

The rotation from the body frame to the inertial frame can be defined with the matrix

$$\boldsymbol{R} = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}$$

in which $S_x = \sin(x)$ and $C_x = \cos(x)$. Note that this rotation matrix is orthogonal and thus the rotation matrix from the inertial frame to the body frame is $\boldsymbol{R^{-1}} = \boldsymbol{R^T}$.

To obtain the angular velocities in the body frame from the angular velocities in the inertial frame, the matrix $\boldsymbol{W_\eta}$ should be used:

$$\boldsymbol{\nu} = \boldsymbol{W}_\eta \dot{\boldsymbol{\eta}}, \qquad \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Its inverse is the transformation matrix from the body frame to the inertial frame, which will be used later to assemble the quadrotor dynamics:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{W}_\eta^{-1}\boldsymbol{\nu}, \quad \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi/C_\theta & C_\phi/C_\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

where $T_x = \tan(x)$.

As shown in Figure 1 and stated in assumption 5, the drone is symmetric with the arms aligned with the body axes. Therefore, the inertia matrix $\mathbf{I}$ is diagonal, and $I_{xx} = I_{yy}$:

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

C. Quadrotor Dynamics

The force f_i created by the angular velocity of rotor i, denoted with ω_i, in the direction of the rotor axis is:

$$f_i = k\omega_i^2$$

Additionally, torque τ_{M_i} is created around the rotor axis:

$$\tau_{M_i} = b\omega_i^2 + I_r\dot{\omega}_i$$

where the lift constant is k, the drag constant is b and the inertia moment of the rotor is I_r. As assumption 4 considered, the rotor's inertia is small, and $\dot{\omega}_i$ is also usually small. Therefore, this term can be omitted.

The rotors create thrust T in the direction of the body z-axis, and torque $\boldsymbol{\tau_B}$ consists of torques $\tau_\phi, \tau_\theta, \tau_\psi$ in the direction of the corresponding body frame angles:

$$T = \sum_{i=1}^{4} f_i = k\sum_{i=1}^{4}\omega_i^2, \qquad \boldsymbol{T}^B = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}$$

$$\boldsymbol{\tau}_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk\left(-\omega_2^2 + \omega_4^2\right) \\ lk\left(-\omega_1^2 + \omega_3^2\right) \\ \sum_{i=1}^{4}\tau_{M_i} \end{bmatrix}$$

in which l is the distance between the rotor and the center of mass of the quadcopter.

The Newton-Euler equations for the quadrotor are:

$$m\ddot{\boldsymbol{\xi}} = \boldsymbol{G} + \boldsymbol{R}\boldsymbol{T}_B$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m}\begin{bmatrix} C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi S_\theta C_\phi - C_\psi S_\phi \\ C_\theta C_\phi \end{bmatrix}$$

where g is Earth's gravity, 9.81 m/s^2. Additionally:

$$\boldsymbol{I}\dot{\boldsymbol{\nu}} + \boldsymbol{\nu} \times (\boldsymbol{I}\boldsymbol{\nu}) + \boldsymbol{\Gamma} = \boldsymbol{\tau_B}$$

$$\dot{\boldsymbol{\nu}} = \boldsymbol{I}^{-1}\left(-\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{bmatrix} - I_r\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\omega_\Gamma + \boldsymbol{\tau_B}\right)$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz})\,qr/I_{xx} \\ (I_{zz} - I_{xx})\,pr/I_{yy} \\ (I_{xx} - I_{yy})pq/I_{zz} \end{bmatrix} - I_r\begin{bmatrix} q/I_{xx} \\ -p/I_{yy} \\ 0 \end{bmatrix}\omega_\Gamma + \begin{bmatrix} \tau_\phi/I_{xx} \\ \tau_\theta/I_{yy} \\ \tau_\psi/I_{zz} \end{bmatrix}$$

where $\omega_\Gamma = \omega_1 - \omega_2 + \omega_3 - \omega_4$. By assumption 4, the term that includes I_r and ω_Γ can be omitted.

Finally, this work considers aerodynamic drag caused by the vehicle's translation. Therefore, the revised Netwon equation is:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m}\begin{bmatrix} C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi S_\theta C_\phi - C_\psi S_\phi \\ C_\theta C_\phi \end{bmatrix} - \frac{1}{m}\begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

where A_x, A_y, and A_z are the drag force coefficients for velocities in the corresponding directions of the inertial frame.

D. State-space Representation

The state of the quadrotor can be represented as follows:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \\ \boldsymbol{\eta} \\ \boldsymbol{\nu} \end{bmatrix}$$

that is, a column vector of $4 \cdot 3 = 12$ components that contains the inertial position, the inertial velocity, the Euler angles and the angular velocities in the body frame. The control inputs are:

$$\boldsymbol{U} = \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}$$

that is, the total thrust T and the torques τ that cause a roll, pitch, and yaw angle change, respectively.

The dynamics previously derived can be assembled in vector $\dot{\boldsymbol{X}} = f(\boldsymbol{X}, \boldsymbol{U})$, which represents the change of the state as a function of the state and the inputs. That is:

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \frac{T}{m}[C_\psi S_\theta C_\phi + S_\psi S_\phi] - \frac{A_x}{m}\dot{x} \\ \frac{T}{m}[S_\psi S_\theta C_\phi - C_\psi S_\phi] - \frac{A_y}{m}\dot{y} \\ -g + \frac{T}{m}[C_\theta C_\phi] - \frac{A_z}{m}\dot{z} \\ p + q[S_\phi T_\theta] + r[C_\phi T_\theta] \\ q[C_\phi] - r[S_\phi] \\ q\frac{S_\phi}{C_\theta} + r\frac{C_\phi}{C_\theta} \\ \frac{I_{yy}-I_{zz}}{I_{xx}}qr + \frac{\tau_\phi}{I_{xx}} \\ \frac{I_{zz}-I_{xx}}{I_{yy}}pr + \frac{\tau_\theta}{I_{yy}} \\ \frac{I_{xx}-I_{yy}}{I_{zz}}pq + \frac{\tau_\psi}{I_{zz}} \end{bmatrix}$$

E. Mathematical Formulation

The problem in hand consists of minimizing the time a drone takes to complete a race, that is, to cross all the gates in a specific order as fast as possible. In this paper, the gates are considered point constraints only, and it does not matter the direction of the velocity vector. This can be formulated mathematically as an optimal control problem:

Minimize $\int_{t_0}^{t_f} 1dt = \int_0^{t_f} 1dt = t_f$

subject to:

$$\dot{\boldsymbol{X}} = f(\boldsymbol{X}, \boldsymbol{U})$$

$$\boldsymbol{X}(0) = 0$$

$$\boldsymbol{x}(t_i) = \boldsymbol{g_i}, \quad t_i < t_{i+1} \text{ for } i = 1..n_{gates} - 1$$

$$z \geq 0$$

$$0 \leq T \leq T_{max}$$

$$|\tau_\phi| \leq \tau_{\phi,max}$$

$$|\tau_\theta| \leq \tau_{\theta,max}$$

$$|\tau_\psi| \leq \tau_{\psi,max}$$

The first constraint imposes the quadrotor dynamics. The second one, states that the initial state of the drone is 0, that is, the drone is stopped in the origin and the body axes match the global axes. The third constraint are the gates: the drone's position $\boldsymbol{x}$ has to match the gates' positions $\boldsymbol{g_i}$ in increasing times (that is, the gates have to be crossed in order, from number 1 to number n_{gates}). The next constraint ensures that the drone will not crash against the ground. The last four constraints limit the control inputs, necessary to account for the limitations of the quadrotor's propellers and motors.

III. PROBLEM IMPLEMENTATION

The problem was implemented in GPOPS-II [2], a MATLAB software to solve optimal control problems. As shown in GPOPS-II's User Guide [3], a program must have three main functions: the main, the continuous and the endpoint function, which are explained in the following subsections.

To specify the gate constraints, the program is divided in n_{gates} phases. The first phase is from the initial state (where all the states are 0) to the first gate, the second phase is from the end of the first phase to the second gate, and so on.

A. Main Function

In this function, the race and drone parameters are loaded, the program is set up and the bounds and guesses are specified:

- **Bounds:** the bounds limit the states and controls allowed for each phase. The time, position, velocity, orientation and rates are limited to realistic values, although the quadrotor could technically surpass them. For example, this work does not allow the drone to fly faster than 50 m/s. As for the control inputs, they are limited to the allowed values as shown in Subsection II-E.
  Additionally, the z is limited to values greater than 0 to avoid the drone crashing into the ground, the initial state is set to 0, the difference of states in contiguous phases is set to 0 and the final position for each phase is set to the corresponding gate. To avoid having redundant constraints, the initial positions of each phase are not set to the previous gate.
- **Guesses:** they set values to the time, state and control for each phase to help the algorithm converge.

Finally, the data is saved and plotted using auxiliary functions.

B. Continuous Function

This function contains all the dynamics explained in Section II. For every phase, $\dot{\boldsymbol{X}}$ is computed and returned. Additionally, the path constraint variable is returned, that is, the state z. As mentioned before, its minimum is set to 0 using the lower and upper path constraint bounds.

C. Endpoint Function

The endpoint function returns the objective function, which is the final time of the last phase, and also returns the difference of the states and times between contiguous phases as something called 'eventgroup'. This difference is set to 0 by the bounds, like previously explained.

D. Video Functions

To make the video of the drone performing the race, the time, position and orientation vectors obtained with GPOPS were saved to an Excel file, which was read by a Python script. Using ROS [4], the script publishes topics from which Gazebo [5] (a simulation environment for ROS) is subscribed. These topics include the position and orientation of the drone, and are published with a period that matches the time difference of every point. A quadrotor model in Gazebo receives the ROS messages and moves and rotates as they indicate, thus giving the impression that the drone is moving. Cuboids were added to the environment to represent the gates.

IV. RESULTS

A. Quadrotor Parameters

The following quadrotor parameters were used:

TABLE I

Quadrotor parameters

Parameter	**Value**	**Unit**
Mass	1.1	kg
I_{xx}	0.03	$kg \cdot m^2$
I_{yy}	0.03	$kg \cdot m^2$
I_{zz}	0.16	$kg \cdot m^2$
A_x	0.10	kg/s
A_y	0.10	kg/s
A_z	0.20	kg/s
T_{max}	25	N
$\tau_{\phi,max}$	5	$-$
$\tau_{\theta,max}$	5	$-$
$\tau_{\psi,max}$	1.5	$-$

B. Experiments

1) Simple experiment without the z constraint: A

2) Simple experiment with z constraint: Show simple experiment with/out considering z constraint. Show gates.

Show gates and plots of final experiment.

The video can be found in ...

V. CONCLUSIONS

Copy abstract... Future work could impose additional gate constraints such as limiting the velocity to a cone, although this constraint is not noticeable if the gates are positioned one (more or less) in front of the other as would happen in a real race.

REFERENCES

[1] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok. A novel ultrathin elevated channel low-temperature poly-Si TFT. 20:569–571, November 1999.

[2] Michael A Patterson and Anil V Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[3] Michael A Patterson and Anil V Rao. GPOPS-II: A general-purpose MATLAB software for solving multiple-phase optimal control problems, 2016.

[4] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[5] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.