We are IntechOpen, the world's leading publisher of Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

Quadrotor Unmanned Aerial Vehicles: Visual Interface for Simulation and Control Development

Manuel A. Rendón

Abstract

Quadrotor control is an exciting research area. Despite last years developments, some aspects demand a deeper analysis: How a quadrotor operates in challenging trajectories, how to define trajectory limits, or how changing physical characteristics of the device affects the performance. A visual interface development platform is a valuable tool to support this effort, and one of these tools is briefly described in this Chapter. The quadrotor model uses Newton-Euler equations with Euler angles, and considers the effect of air drag and propellers' speed dynamics, as well as measurement noise and limits for propeller speeds. The tool is able to test any device just by setting a few parameters. A three-dimensional optimal trajectory defined by a set of waypoints and corresponding times, is calculated with the help of a *Minimum Snap Trajectory* planning algorithm. Small Angle Control, Desired Thrust Vector (DTV) Control and Geometric Tracking Control are the available strategies in the tool for quadrotor attitude and trajectory following control. The control gains are calculated using *Particle Swarm Optimization*. Root Mean Square (RMS) error and Basin of Attraction are employed for validation. The tool allows to choose the control strategy by visual evaluation on a graphical user interface (GUI), or analyzing the numerical results. The tool is modular and open to other control strategies, and is available in GitHub.

Keywords: Quadrotor, Trajectory Planning, Trajectory Tracking

1. Introduction

Quadrotors are a special type of unmanned aerial vehicles (UAVs), increasingly employed last years for mapping, surveillance, searching and tracking operations, in rescue missions, agriculture, traffic management, landscape film making, and others [1–3].

When quadrotors applications demand large angle attitude control and obstacle avoidance, the following areas still need to be strengthened: Aggressive maneuvering control, visual-based control, localization in indoor environments, optimizing the computational cost for complex algorithms, and fault-tolerant disturbance rejection [4]. Several controllers have been proposed for these tasks: classic techniques as PD [5] and PID [6], optimal techniques as LQR [7] and LQG [8], non-

linear techniques such as Lyapunov [9] and *Backstepping* [5, 9] and intelligent and adaptive control techniques as Fuzzy200 [10] and Reinforcement Learning [11].

Research in quadrotor demand sophisticated equipment and costly laboratory. However, it can be optimized employing low cost virtual development tools. In autonomous control the path planning, path tracking and joint operation with other UAVs, can be supported by optimization techniques with support of software tools [1, 2, 12]. Published works relate the use of software such as Visual Basic, MatLab, Panda3D, Gazebo, or more open applications developed in languages as Python and C++ [2]. In [1], a UAV 3D flight environment programmed in Python and developed in Panda3D is presented. A GUI developed in LabVIEW was published in [2], and other developed on MatLab-Simulink employs quadratic linear regulator (LQR) control [13].

Gazebo is an important virtual environment for robotics. Its integration with ROS provides a powerful testbed to analyze control algorithms. In [12] works that employ Gazebo and ROS for developing simulation of UAVs are described.

Most of the cited simulation tools have a difficult start for users with little programming experience. Even open-source models can be tricky. An interesting tool depicted in [14, 15] gives support for quadrotor control development, analyzing and comparing various control strategies on challenging trajectories. It may be used by beginning users with not much knowledge in ROS, Gazebo or in programming languages such as Python or C ++. The tool easies the understanding of quadrotor dynamics and related equations, as well as the development of control strategies. The present Chapter describes this user-friendly framework, the employed techniques are described in [14–17].

Section 2 presents a description of the employed model. Section 3 explains the optimal trajectory planning method. Section 4 describes the controllers available in the tool. Section 5 presents the graphical user interface developed by the tool. Section 6 presents the graphical and numerical results. In the end of the Chapter, the Section 7 emphasizes the main aspects and critical issues related.

2. Quadrotor model

The equations are described in the rigid body model of the quadrotor, and its displacement is related to an inertial *frame*, fixed on the Earth surface [18]. Three main *frames* are considered for a quadrotor model: Inertial (**I**), Vehicle (**V**) and Rigid Body (**B**), as illustrated in **Figure 1**.

The quadrotor owns 4 propellers in cross configuration. Each pair of propellers (1,3) and (2,4) rotates in opposite directions (**Figure 1**). Setting different rotor speeds to each pair ($\omega_2 \neq \omega_4$) or ($\omega_1 \neq \omega_3$) produces *roll* or *pitch* rotations with corresponding lateral motion. *Yaw* rotation results from rolling moments difference ($M_1 + M_3 - M_2 - M_4$) between propellers [9]. Maximum and minimum motor speeds are some of the parameters to be set in the model.

Newton-Euler equations describe quadrotor dynamics and kinematics [9, 18]. The Rotation Matrix represents the rotation around the three axes in the sequence $Z - X - Y$ (1), and is described in (2).

$$\mathbf{R}^{\mathbf{I}}_{\mathbf{B}_{\mathrm{Z-X-Y}}} = \mathbf{R}^{\mathbf{I}}_{\mathbf{A2}}(\psi)\mathbf{R}^{\mathbf{A2}}_{\mathbf{A1}}(\phi)\mathbf{R}^{\mathbf{A1}}_{\mathbf{B}}(\theta) \tag{1}$$

$$\mathbf{R}^{\mathbf{I}}_{\mathbf{B}_{\mathrm{Z-X-Y}}} \triangleq \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\theta s\phi c\psi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \tag{2}$$

The angle ϕ around the x axis is the *roll angle*, the angle θ around the y axis is *pitch angle*, and the ψ angle around the z axis is *yaw angle*.

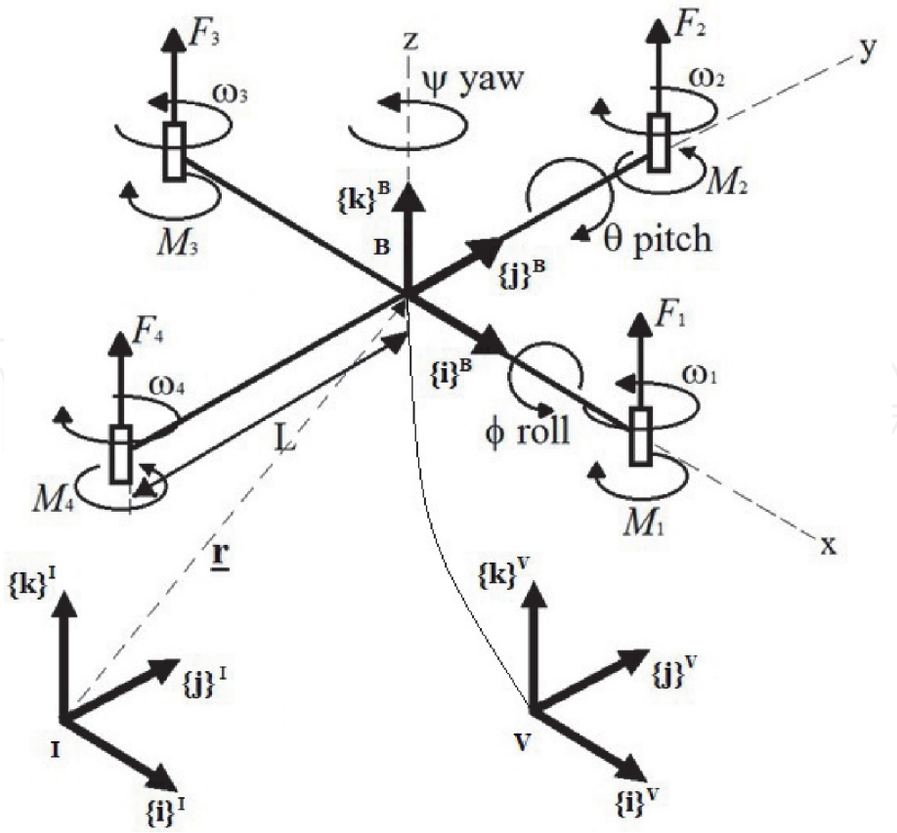

Figure 1.
Main frames in a quadrotor.

The orientation vector is defined by η and the position of quadrotor's center of mass is defined in the reference frame **I** by **r**. Angular velocities in the body frame **B** (p, q and r) are defined by ν (3).

$$\eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \nu = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3}$$

The derivative of the angles ϕ, θ and ψ and the angular velocities measured by a sensor fixed to the *Frame* of the Rigid Body are not the same. p, q and r are the angular velocities around the x, y and z axes of the Rigid Body *Frame*. The relationship with the angular rates $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ in the same *frame* **B** is in (4).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{T} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{4}$$

Considering that the body is symmetrical with respect to the x-z and y-z planes of the *frame* **B**,and that the only forces acting on it are the weight and the four thrusts, its resulting linear acceleration with respect to the inertial *Frame* can be described using Newton's Second Law. Air drag forces are represented by a matrix in (5), and the values of A_x, A_y and A_z, are inputs on the parameters set.

$$m\{\ddot{\mathbf{r}}\}^{\mathrm{I}} = \left\{ \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \right\}^{\mathrm{I}} + \mathbf{R}_{\mathbf{B}}^{\mathrm{I}} \left\{ \begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} - \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix} \right\}^{\mathrm{B}} \tag{5}$$

Besides the force each rotor produces a moment in the rigid body perpendicular to the plane of rotation of the propeller (M_i), contrary to the direction of rotation of the propellers. L is the size of the quadrotor arm (**Figure 1**) and **J** the inertia matrix presented in (6).

$$\mathbf{J} = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \tag{6}$$

Due to the forces produced by the rotors, moments are produced in the rigid body ($L \cdot F_i$), causing the system to rotate around the x and y axes. The rotation around the z axis is due to the torque created by the rotation of the motors, which are fixed to the plant. Based on the Coriolis Equation, the equation that describes the angular acceleration in the *Frame* **B** is in (7).

$$\mathbf{J} \left\{ \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \right\}^{\mathrm{B}} = \left\{ \begin{bmatrix} L(F_2 - F_4) \\ L(F_1 - F_3) \\ M_1 + M_2 + M_3 + M_4 \end{bmatrix} \right\}^{\mathrm{B}} - \left\{ \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right\}^{\mathrm{B}} \times \mathbf{J} \left\{ \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right\}^{\mathrm{B}} \tag{7}$$

For simulating the measurement noise, the tool allows to add a random signal to position and orientation variables and their derivatives. A first order delay between the comands u_1 and $\mathbf{u_2}$ and rotor speed variations, with a time constant τ seconds, was included in the tool to simulate the rotor dynamics.

3. Trajectory planning for a set of waypoints

Due to the low inertia of quadrotor it is necessary to calculate a smooth trajectory to minimize the risk of collapse. Euler–Lagrange equations are used to find the minimum *snap* trajectory [15, 19]. For a two waypoints trajectory, the boundary conditions of position, velocity, acceleration, and *jerk* are defined in **Table 1**.

With this boundary conditions the equations' coefficients for the two-points optimal desired trajectory are calculated for each coordinate of position (x_{des}, y_{des} and z_{des}) and orientation (ψ_{des}) (8).

$$\begin{bmatrix} x_{des} \\ y_{des} \\ z_{des} \\ \psi_{des} \end{bmatrix} = \begin{bmatrix} c_{1,7} & c_{1,6} & c_{1,5} & c_{1,4} & c_{1,3} & c_{1,2} & c_{1,1} & c_{1,0} \\ c_{2,7} & c_{2,6} & c_{2,5} & c_{2,4} & c_{2,3} & c_{2,2} & c_{2,1} & c_{2,0} \\ c_{3,7} & c_{3,6} & c_{3,5} & c_{3,4} & c_{3,3} & c_{3,2} & c_{3,1} & c_{3,0} \\ c_{4,7} & c_{4,6} & c_{4,5} & c_{4,4} & c_{4,3} & c_{4,2} & c_{4,1} & c_{4,0} \end{bmatrix} \begin{bmatrix} t^7 \\ t^6 \\ t^5 \\ t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \tag{8}$$

Time t	Position $x_{des}(t)$	Velocity $\dot{x}_{des}(t)$	Acceleration $\ddot{x}_{des}(t)$	Jerk $\dddot{x}_{des}(t)$
0	$x_{des}(0)$	0	0	0
T	$x_{des}(T)$	0	0	0

Table 1.
Boundary conditions.

Desired angles for *roll* (ϕ_{des}) and *pitch* (θ_{des}) are calculated from ψ_{des}, the equations depend on which control strategy is employed.

This procedure yields a minimum *snap* trajectory for two points. For additional waypoints it is necessary to consider more equations and intermediary restrictions [15].

The tool calculates the complete optimized trajectory for any set of waypoints. Desired trajectory $\mathbf{r}_{des}$, orientation ψ_{des}, and their derivatives are calculated for being used in the control algorithms.

4. Attitude and trajectory following control strategies

Some of the challenges to be overcome in quadrotor operation are: attitude stability for large angles, trajectory following, collision avoidance through aggressive maneuvers, monitoring, and others [2, 4].

The control architecture employed in the following control strategies uses two cascaded loops. The inner loop (attitude control) runs in a fast time-scale and is assumed exponentially stable. The outer loop (position control) runs in a slow time-scale, with a higher bandwidth [4]. All of them employ a feed-forward with proportional plus derivative structure (FF-PD). The tool is open to easily add more control strategies.

4.1 Small angle control

Small Angle control assumes an operation not too far from the hovering condition. A simple heuristic method with FF-PD control calculates the required accelerations for the desired trajectory (9).

$$\begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c \\ \ddot{z}_c \end{bmatrix} = \mathbf{r}_{des} + K_p\,\mathbf{e}_r + K_d\,\mathbf{e}_r \tag{9}$$

It is assumed small deviations from zero in *roll* and *pitch* angles, small deviations in the *yaw* angle from the desired value, and angular velocities close to zero. The algorithm assumes all upward-pointing thrust vectors (control signal u_1 in (10)).

$$u_1 = m(g + \ddot{z}_c) \tag{10}$$

After linear simplifications the equations for attitude control are defined in (11) and (12) [16].

$$\eta_c = \begin{bmatrix} \phi_c \\ \theta_c \\ \psi_c \end{bmatrix} = \begin{bmatrix} \frac{1}{g}\left(\ddot{x}_c \sin(\psi_{des}) - \ddot{y}_c \cos(\psi_{des})\right) \\ \frac{1}{g}\left(\ddot{x}_c \cos(\psi_{des}) + \ddot{y}_c \sin(\psi_{des})\right) \\ \psi_{des} \end{bmatrix} \tag{11}$$

$$\nu_c = \begin{bmatrix} p_c \\ q_c \\ r_c \end{bmatrix} = \mathbf{T}\,\eta_c \tag{12}$$

A PD control law is used for attitude control and to calculate $\mathbf{u}_2$ (13) [16].

$$\mathbf{u}_2 = K_R(\eta_c - \eta) + K_\nu(\nu_c - \nu) \tag{13}$$

4.2 Desired thrust vector control

At high speeds and *roll* and *pitch* angles far away from zero, a more robust strategy is necessary. A FF-PD control law calculates the control signal u_1 for the trajectory following and compensates the gravity. Vector $\mathbf{t}$ is calculated with the attitude for the desired effect (14). As u_1 acts along the z direction in frame $\mathbf{B}$ (axis $\mathbf{b}_z$), it must be referred to frame $\mathbf{I}$ (15).

$$\mathbf{t} = m\left(\mathbf{r}_{des} + K_p\,\mathbf{e}_r + K_d\,\mathbf{e}_r\right) + mg\mathbf{a}_z \tag{14}$$

$$u_1 = \mathbf{t}^T\mathbf{R}\,\mathbf{b}_z \tag{15}$$

The axis $\mathbf{b}_z$ is desired to be aligned with $\mathbf{t}$. The desired rotation matrix $\mathbf{R}_{des}$ is calculated from the equation of rotation of $\mathbf{b}_z$ in the direction of $\mathbf{t}$ (16).

$$\mathbf{R}_{des}\mathbf{b}_z = \frac{\mathbf{t}}{\|\,\mathbf{t}\,\|} \tag{16}$$

As we know ψ_{des} we use (16) to calculate ϕ_{des} and θ_{des}. $\mathbf{R}_{des}$ is constructed with these three angles with the same structure of (2). The error in rotation $\Delta\mathbf{R}$ is calculated from (17).

$$\Delta\mathbf{R} = \left(\mathbf{R}_{\mathbf{B}}^{\mathrm{I}}\right)^T\mathbf{R}_{des} \tag{17}$$

With the Rodrigues formula [20], the axis of rotation $\mathbf{v}$ and the rotation angle β are calculated (18). $\mathbf{I}_{3\times3}$ is the identity matrix and $\hat{\mathbf{v}}$ is the skew-symmetric matrix of $\mathbf{v}$ (19). The related error is calculated with (20).

$$\Delta\mathbf{R} = \mathbf{I}_{3\times3}\,\cos\beta + \mathbf{v}\mathbf{v}^T(1-\,\cos\beta) + \hat{\mathbf{v}}\,\sin\beta \tag{18}$$

$$\hat{\mathbf{v}} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{19}$$

$$\mathbf{e}_R = \beta\mathbf{v} \tag{20}$$

A PD control law is used to calculated $\mathbf{u}_2$ (22).

$$\mathbf{e}_\nu = \nu_c - \nu \tag{21}$$

$$\mathbf{u}_2 = \nu \times \mathbf{J}\nu + \mathbf{J}(-K_R\,\mathbf{e}_R - K_\nu\,\mathbf{e}_\nu) \tag{22}$$

Basin of attraction Ψ limits the set of rotations from which the quadrotor is able to converge to the hovering state. It is a dimension of the set of angular and linear velocities for a stable performance. For this control strategy it must be lower than 2 (23).

$$\Psi = \mathrm{tr}\left(\mathbf{I}_{3\times3} - \mathbf{R}_{des}^T\mathbf{R}_{\mathbf{B}}^{\mathrm{I}}\right) < 2 \tag{23}$$

4.3 Geometric tracking control

Geometric Tracking Control exhibits almost global exponential attractiveness to the zero equilibrium of tracking errors [17]. $\mathbf{t}$ and u_1 are calculated as in (14) and (15). $\mathbf{b}_x$ is the desired direction vector in the first body-fixed axis (24).

$$\mathbf{b}_x = \begin{bmatrix} \cos\psi_{des} \\ \sin\psi_{des} \\ 0 \end{bmatrix} \tag{24}$$

With $\mathbf{t}$ and $\mathbf{b}_x$ is possible to calculate $\mathbf{R}_{des}$ with (25) to (27) [17].

$$\mathbf{b}_z = \frac{\mathbf{t}}{\|\mathbf{t}\|} \tag{25}$$

$$\mathbf{b}_y = \frac{\mathbf{b}_z \times \mathbf{b}_x}{\|\mathbf{b}_z \times \mathbf{b}_x\|} \tag{26}$$

$$\mathbf{R}_{des} = \begin{bmatrix} \mathbf{b}_y \times \mathbf{b}_z & \mathbf{b}_y & \mathbf{b}_z \end{bmatrix} \tag{27}$$

(27) calculates the desired attitude for the quadrotor given $\mathbf{t}$ and ψ_{des}. The basin of attraction Ψ (28) is bigger than in the previous strategy (23) as this is a more robust approach [17].

$$\Psi = \frac{1}{2}\,\mathrm{tr}\left(\mathbf{I}_{3\times3} - \mathbf{R}_{des}^T \mathbf{R}_{\mathbf{B}}^{\mathrm{I}}\right) < 2 \tag{28}$$

Attitude tracking error and angular velocity error are calculated from (29, 30). The control vector $\mathbf{u}_2$ is calculated in (31) [17].

$$\mathbf{e}_R = \frac{1}{2}\left(\mathbf{R}_{des}^T \mathbf{R}_{\mathbf{B}}^{\mathrm{I}} - \mathbf{R}_{\mathbf{B}}^{\mathrm{I}\,T} \mathbf{R}_{des}\right)^{\vee} \tag{29}$$

$$\mathbf{e}_\nu = \nu - \mathbf{R}_{\mathbf{B}}^{\mathrm{I}\,T} \mathbf{R}_{des}\, \nu_{des} \tag{30}$$

$$\mathbf{u}_2 = \nu \times \mathbf{J}\nu + \mathbf{J}(-K_R \mathbf{e}_R - K_\nu \mathbf{e}_\nu) - \mathbf{J}\left(\hat{\nu}\, \mathbf{R}_{\mathbf{B}}^{\mathrm{I}\,T} \mathbf{R}_{des}\, \nu_{des} - \mathbf{R}_{\mathbf{B}}^{\mathrm{I}\,T} \mathbf{R}_{des}\, \dot{\nu}_{des}\right) \tag{31}$$

4.4 Particle swarm optimization for control gains tuning

A PSO algorithm is employed to tune the control gains. Some adjustments were performed to reduce the processing [14, 15], s. Each particle α_i is a vector with the proportional and derivative gains (32).

$$\alpha_i = \begin{bmatrix} K_p & K_d & K_R & K_\nu \end{bmatrix} \tag{32}$$

Using a predefined set of waypoints chosen by the user, the code calculates the desired trajectory and tests each particle, calculating the RMS error. The PSO algorithm evolves in the direction of the best validated solution in each iteration, until achieving a minimum error tolerance. For faster convergence, every time a particle is evaluated, the evaluation is interrupted in the middle of the trajectory if the error increases above a predefined limit.

5. The graphical user interface

A user-friendly 3D animated GUI was developed in MatLab. It is able to evaluate and compare the performance of various quadrotor control strategies for any user-chosen trajectory. A few of quadrotor parameters are easily set in this interface [15].

A red vertical line indicates the upper side of the device, and a red circle the front rotor. Waypoints are represented by red markers, the desired trajectory is on

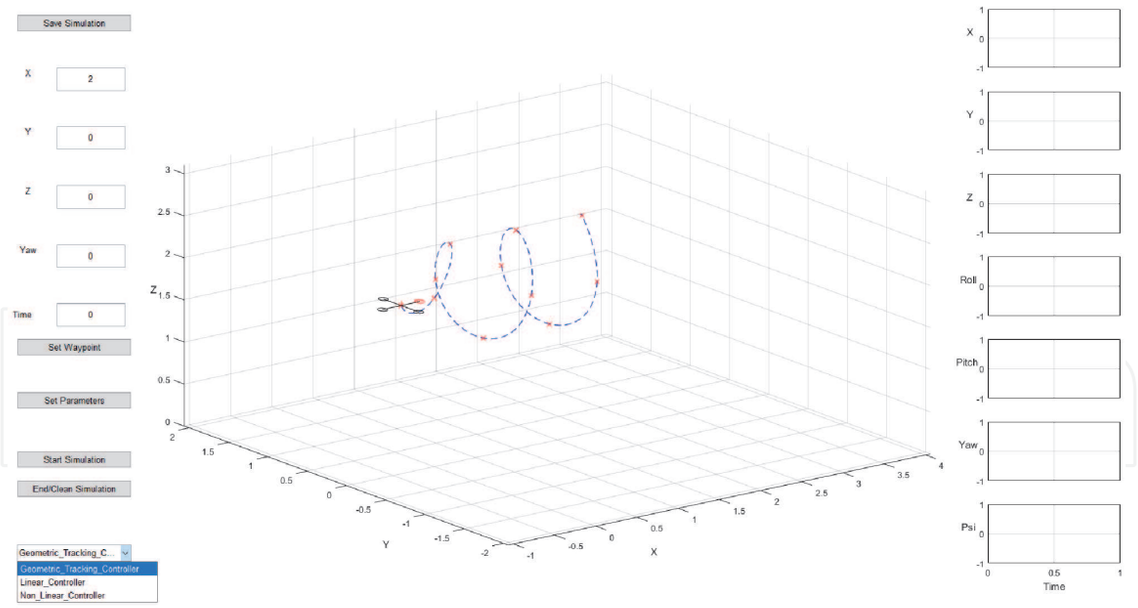

Figure 2.
Graphical user interface.

dashed blue line, and the 3D interface axis limits are automatically adjusted from the waypoints (**Figure 2**).

Waypoints, quadrotor parameters and simulation comands are set in th buttons on the left side of the GUI.

The user may test its own control strategy just by creating the corresponding code like the following example:

```
function New_Controller
% declare global variables
global quad;
% holds the quadrotor in the last waypoint
if(quad.iteracao > length(quad.rdes(1,:)))
Controlador_Position_Hold()
end
% start the controller code and calculate the global control commands
quad.u1=...
quad.u2=...
```

The code New_Controller.m must be stored in the folder Controllers, and will be recognized in the dropdown menu in the lower left side of **Figure 2**.

6. Results

The tool developed in MatLab and is available in GitHub [21]. The values employed for these results are in **Table 2** [3].

For simulating the measurement noise, a random signal was added to position ($\pm 0.01\,m$) and orientation ($\pm 0.5^{\circ}$) variables and their derivatives. Maximum and minimum motor speeds make the simulation more reliable. A time constant τ of 0.1 s for dynamic rotor speed variation was considered.

Two sets of waypoints of challenging trajectories were tested, each one was simulated in three different initial conditions: Case 1 starting with an orientation

Parameter	Value	Units
J_{xx}	4.856×10^{-3}	$kg.m^2$
J_{yy}	4.856×10^{-3}	$kg.m^2$
J_{zz}	8.801×10^{-3}	$kg.m^2$
k	2.980×10^{-6}	$N.s^2/rad^2$
g	9.81	m/s^2
m	0.468	kg
A_x	0.25	kg/s
A_y	0.25	kg/s
A_z	0.25	kg/s
b	1.100×10^{-7}	$N.m.s^2/rad^2$
L	0.225	m
$N_{i\,max}$	8500	rpm
$N_{i\,min}$	1300	rpm
τ	0.1	s

Table 2.
Quadrotor parameters [3].

angle of $\phi(0) = 0^\circ$, Case 2 with $\phi(0) = 88^\circ$, and Case 3 with $\phi(0) = 178^\circ$. The three control strategies available in the tool were graphically and numerically validated.

6.1 Elliptical helix trajectory

The first trajectory waypoints (**Table 3**) describe an elliptical helix in a forward trajectory along with the x axis of inertial frame $\mathbf{i}_x$.

$t(s)$	$x_{\text{des}}(m)$	$y_{\text{des}}(m)$	$z_{\text{des}}(m)$	$\psi_{\text{des}}(\text{rad})$
0.00	−0.40	0.00	2.00	0.00
1.20	0.00	0.00	2.00	0.00
1.80	0.20	0.00	2.60	0.00
2.40	0.40	0.40	2.00	1.57
3.00	0.60	0.00	1.40	3.14
3.60	0.80	−0.40	2.00	4.71
4.20	1.00	0.00	2.60	6.28
4.80	1.20	0.40	2.00	7.85
5.40	1.40	0.00	1.40	9.42
6.00	1.60	−0.40	2.00	11.00
6.60	1.80	0.00	2.60	12.57
7.20	1.80	0.00	2.60	12.57

Table 3.
Waypoints test 1.

With this set it was calculated the desired optimal trajectory. Using this trajectory the PSO algorithm calculates the optimal gains for each of the three evaluated control algorithms.

With this gains the simulation was performed and validation errors in each of the four coordinates were registered. **Tables 4–6** present the gains used in each case, and corresponding validation errors.

When calculating the gains using PSO it was observed that the processing time increases substantially with the initial orientation angle ($\phi(0)$). It was also observed in cases 2 and 3 that once a set of optimal gains is calculated, later results do not reduce substantially the validation error.

Small angle control is more sensible to variations in rotational positions as observed in the results. Optimal small angle control gains are bigger than rotational (**Tables 4–6**). The opposite occurs for DTV and geometric tracking.

Case 3 is the most challenging since the quadrotor starts almost upside down. Geometric tracking controller is more sensitive to noise error, PSO did not succeed in finding a set of gains in Case 3.

A visual validation is possible using the tool. Graphics in **Figure 3** show the performance of small angle control in Case 3. The quadrotor drops and recoverers the vertical position, and continues along with the desired trajectory. Graphics on the right supports a visual validation of position and orientation variables.

In the graphics of propellers' speeds of **Figure 4**, dashed red lines indicate the speed extremes. In extreme situations, the controller attains the propellers' limits.

Figure 5 presents the performance of DTV Control on Case 3. It is observed a faster reaction and a more accurate trajectory following.

Near the third point the trajectory leads the quadrotor to its limits. Propellers' speeds in **Figure 6** show a more stable performance than the previous strategy.

Control Strategy	K_p	K_d	K_R	K_ν	x_v %	y_v %	z_v %	ψ_v %
Small Angle	17.96	7.11	0.231	0.16	1.373	7.763	6.301	1.0300
DTV	18.91	10.77	44.920	23.43	2.112	4.907	5.300	0.6455
Geometric Tracking	7.34	7.35	90.590	20.75	3.172	8.807	26.680	0.3973

Table 4.
Elliptical helix trajectory case 1, $\phi(0) = 0°$.

Control Strategy	K_p	K_d	K_R	K_ν	x_v %	y_v %	z_v %	ψ_v %
Small Angle	22.76	2.00	0.765	0.197	2.01	32.69	26.22	0.6344
DTV	19.70	4.85	155.400	26.710	1.58	22.03	27.65	0.4949
Geometric Tracking	8.24	5.62	220.200	27.670	3.17	35.94	51.57	0.4027

Table 5.
Elliptical helix trajectory case 2, $\phi(0) = 88°$.

Control Strategy	K_p	K_d	K_R	K_ν	x_v %	y_v %	z_v %	ψ_v %
Small Angle	8.180	2.349	2.45	0.4371	5.988	64.98	146.1	1.223
DTV	7.268	3.434	236.40	31.4300	4.709	33.71	105.4	1.351
Geometric Tracking	—	—	—	—	—	—	—	—

Table 6.
Elliptical helix trajectory case 3, $\phi(0) = 178°$.

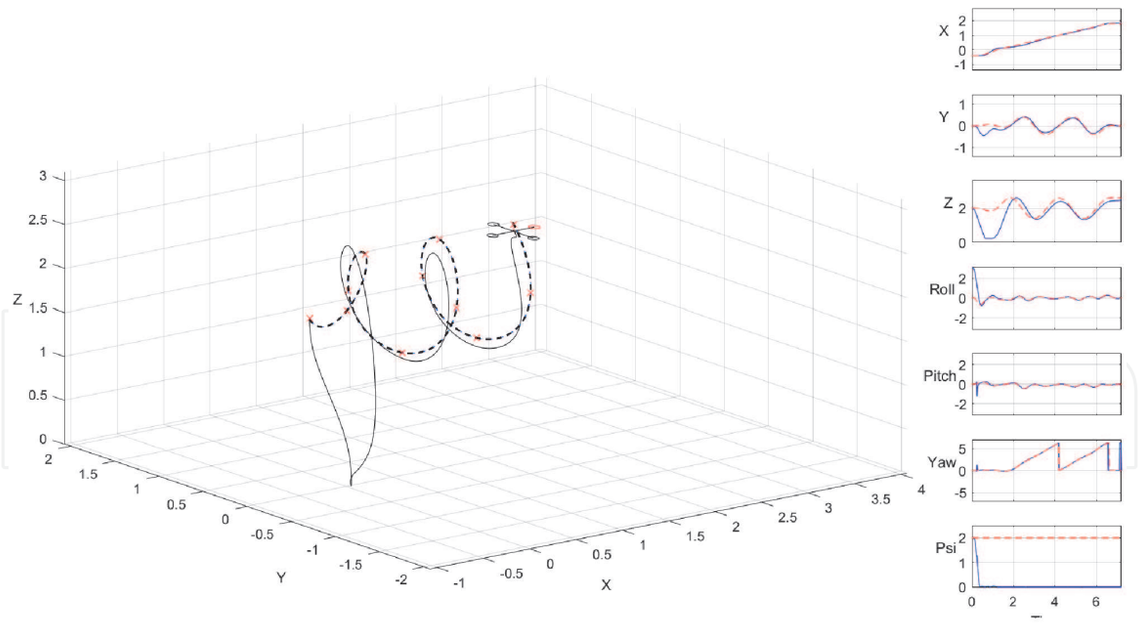

Figure 3.
Elliptical helix trajectory case 3 small angle control, $\phi(0) = 178°$.

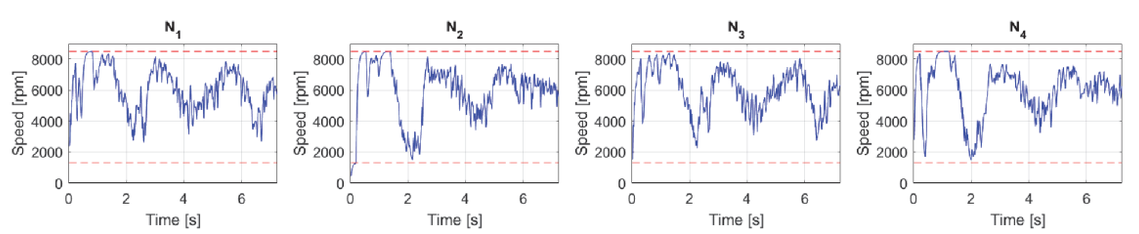

Figure 4.
Propellers' speeds in rpm with small angle control.

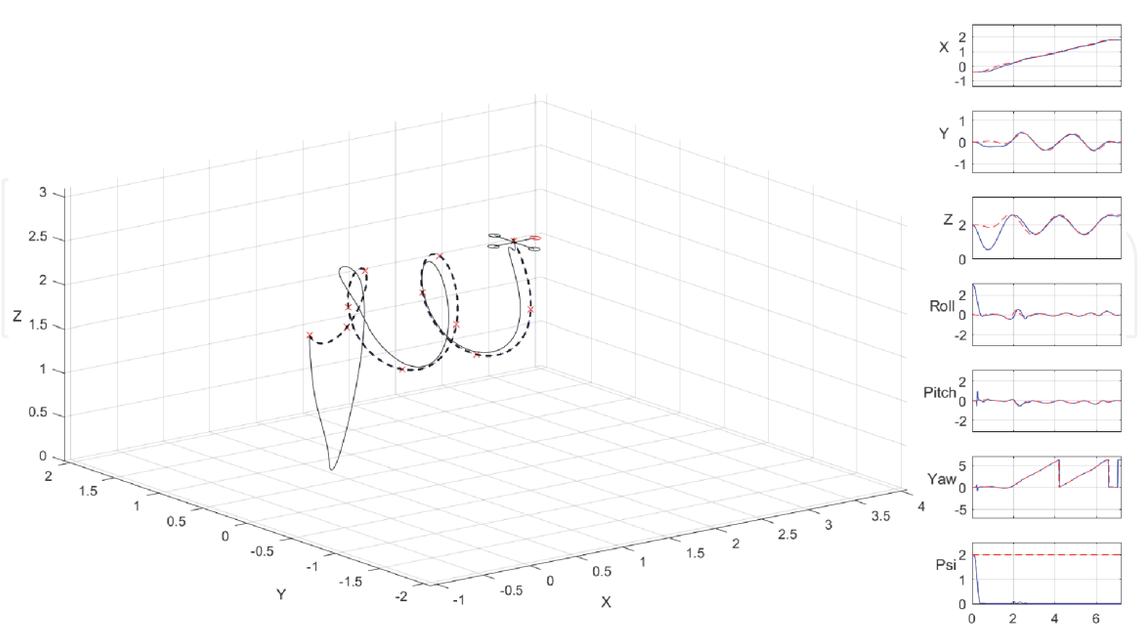

Figure 5.
Elliptical Helix Trajectory Case 3 DTV Control, $\phi(0) = 178°$.

Compared with small angle, this strategy seems to be more robust and accurate.

Figure 7 presents the performance of geometric tracking control in Case 2. This strategy reacts faster to challenging situations, since it recovers in a shorter

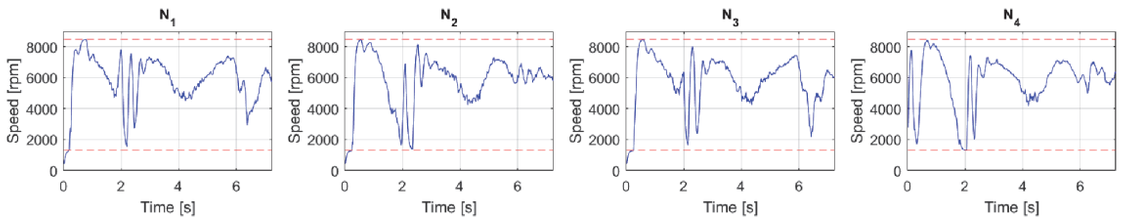

Figure 6.
Propellers' speeds in rpm with DTV control.

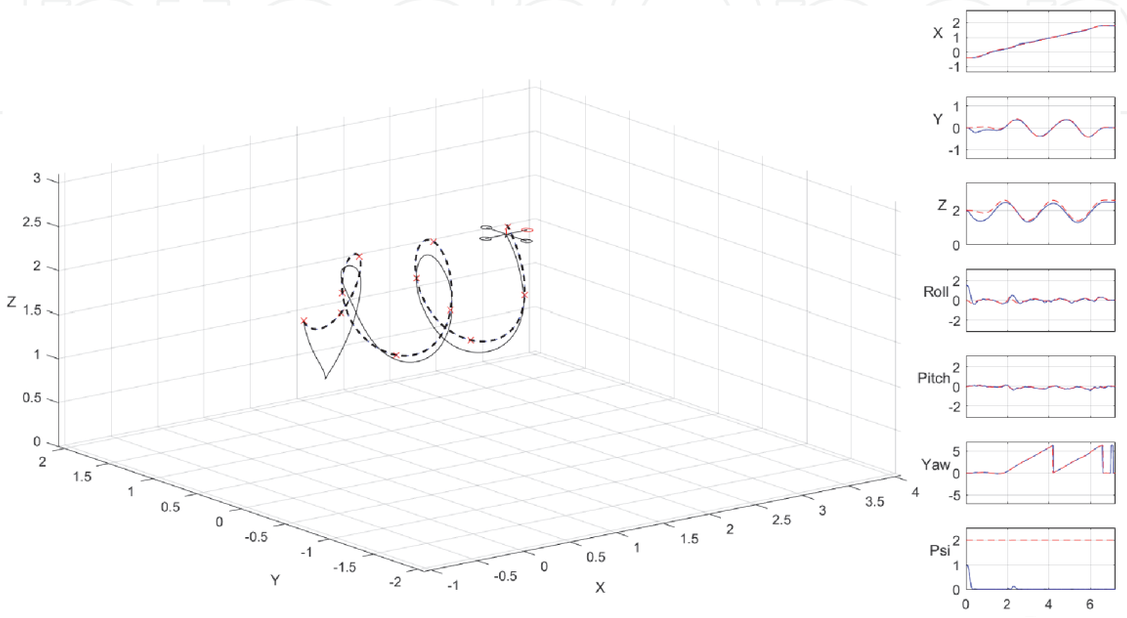

Figure 7.
Elliptical helix trajectory case 2 geometric tracking, $\phi(0) = 88°$.

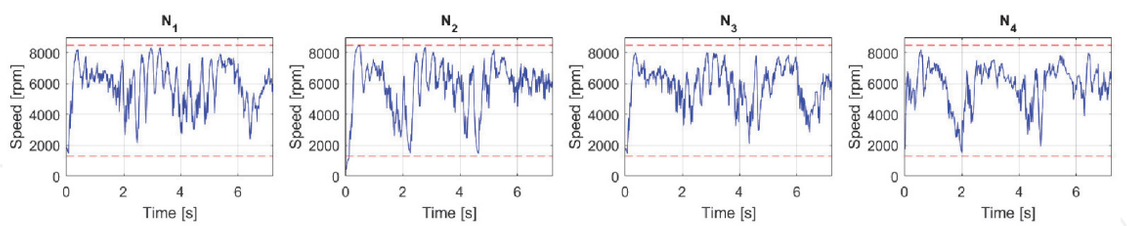

Figure 8.
Propellers' speeds in rpm with geometric tracking.

time its hovering orientation. This fast reaction makes it more sensitive to measuring noise.

Propellers' speeds in **Figure 8** confirm the faster command reaction of the controller compared with **Figures 4** and **6**. This strategy is also sensitive to nonlinear behaviour. When it reaches the propeller limits, it is highly prone to instability.

The gains calculated for the more challenging Cases were later tested on Cases 1 and 2. Despite being less optimal they displayed a stable behaviour. The performance is presented in **Table 7**.

Graphical results of small angle control performance on Case 1, when using the optimal gains compared with the performance with the gains calculated for Case 3. The system holds the attitude performance. The same comparison was performed for DTV control. The decrease in performance is lower than with the small angle controller [15].

Control Strategy	$\phi(0)^{\circ}$	x_v %	y_v %	z_v %	ψ_v %
Small Angle	0°	1.373	7.763	6.301	1.0300
Small Angle[1]	0°	3.111	21.960	28.390	0.2430
Small Angle	88°	2.010	32.690	26.220	0.6344
Small Angle[1]	88°	3.242	51.760	49.580	0.3378
DTV	0°	2.112	4.907	5.300	0.6455
DTV[1]	0°	4.625	15.650	13.310	0.2350
DTV	88°	1.584	22.030	27.650	0.4949
DTV[1]	88°	4.886	41.580	44.530	0.3509
Geometric Tracking	0°	3.172	8.807	26.680	0.3973
Geometric Tracking[2]	0°	3.338	5.917	28.660	0.2615

[1] *Calculated with Gains of Case 3.*
[2] *Calculated with Gains of Case 2.*

Table 7.
Validation comparison for cases 1 and 2.

6.2 Lemniscate Shape Trajectory

A second set of waypoints presented in **Table 8**, depicts a lemniscate shape trajectory varying the position in Z, orienting the front of quadrotor (*yaw* angle ϕ) in the direction of displacement.

Figure 9 depicts the second trajectory.

$t(s)$	$x_{des}(m)$	$y_{des}(m)$	$z_{des}(m)$	ψ_{des}(rad)
0.00	0.00	0.00	2.00	−1.57
0.40	0.12	−0.28	1.58	−0.79
0.80	0.40	−0.40	1.40	0.00
1.20	0.68	−0.28	1.58	0.79
1.60	0.97	0.00	2.00	0.79
2.00	1.25	0.28	2.42	0.79
2.40	1.53	0.40	2.60	0.00
2.80	1.81	0.28	2.42	−0.79
3.20	1.93	0.00	2.00	−1.57
3.60	1.81	−0.28	1.58	−2.36
4.00	1.53	−0.40	1.40	−3.14
4.40	1.25	−0.28	1.58	−3.93
4.80	0.97	0.00	2.00	−3.93
5.20	0.68	0.28	2.42	−3.93
6.60	0.40	0.40	2.60	−3.14
6.00	0.12	0.28	2.42	−2.36
6.40	0.00	0.00	2.00	−1.57

Table 8.
Waypoints test 2.

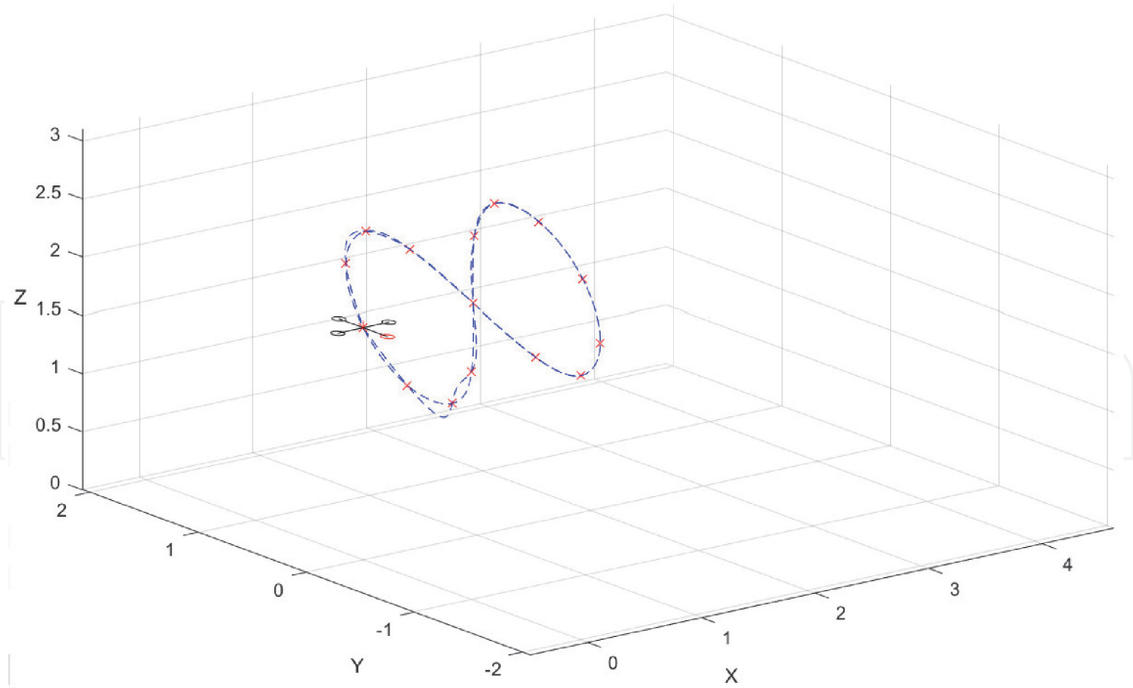

Figure 9.
Lemniscate shape trajectory.

As with the to previous trajectory, the small angle control compensates its limitations with lower gains to guarantee stability, gains are bigger for position control than for attitude.

DTV Control is again the best validated strategy. Geometric tracking presents a lower performance due to its sensitivity to noise and nonlinear operation conditions. The DTV control had presented the best validation even in the most challenging conditions.

Detailed results and analysis of this trajectory are available in [15].

Using the tool the graphical and numerical analysis was easier. Not much programming knowledge was necessary for using and configuration, just basic MatLab programming. Access to quadrotor parameters of the analyzed device, analyzing the influence of measurements noise, and comparing and simulating different control strategies is easier than with other visual platforms. Graphical and numerical simulation results are easily available with the interface buttons.

7. Summary

Quadrotor control is a fascinating research area, but the equations involved and programming skills requirements can be arduous for initiating students. It is a worth to develop motivational appliances for beginners. This was the motivation to present a beginner-friendly visual interface tool for the development of quadrotor control strategies. It is easy to understand, device characteristics are simple to configure, and control algorithms can be implemented and analyzed with not much effort. It is not necessary to have a deep knowledge in programming languages, and may be an introduction to this field of research.

This tool uses RMS and basin of attraction for numerical validation, and the GUI may help to evaluate stability, robustness, and accuracy. It integrates these criteria in a unique interface and helps to measure and visualize details and requirements that may not be so clear using other visual tools.

Baseline controllers are offered so students may compare performance, and is open to easily introduce other strategies for comparison. A trajectory planning based on minimum *snap* give support to trajectory following control, and GUI allows the evaluation in dimensions of position and time.

The tool makes easier and faster to realize critical quadrotor requirements and limitations for challenging applications. These requirements may be related with the complexity of a defined trajectory, the weakness of a control strategy, or the improvements that may be carried out in the quadrotor (size, weight, propeller power, etc.) to accomplish the desired results.

Other controllers can be studied and compared using this tool, such as *Backstepping* and intelligent strategies. In the trajectory planning stage, applications with obstacles may be simulated. Support for multiple quadrotors, communication with the controller via Robotics Operating System (ROS) and implementation of obstacles are some of the future improvements planned for this tool.

Abbreviations

ROS	*Robot Operating System*
PSO	*Particle Swarm Optimization*
DTV	*Desired Thrust Vector*
RMS	*Root Mean Square*
GUI	*Graphical User Interface*
UAV	*Unmanned Aerial Vehicles*
PD	*Proportional plus Derivative Control*
PID	*Proportional Integral Derivative Control*
LQR	*Linear Quadratic Regulator*
LQG	*Linear Quadratic Gaussian*
FF-PD	*Feed-forward Proportional Derivative Control*

Nomenclature

I	Inertial frame
V	Vehicle frame
B	Rigid Body frame
ω_i	Speed of the rotor i
M_i	Moment produced by the rotor i
F_i	Propulsion force produced by the rotor i
ϕ	*Roll*
θ	*Pitch*
ψ	*Yaw*
$\dot{\phi}$	Rate of change of *Roll*
$\dot{\theta}$	Rate of change of *Pitch*
$\dot{\psi}$	Rate of change of *Yaw*
$s\phi, s\theta, s\psi$	Sine of angles ϕ, θ and ψ
$c\phi, c\theta, c\psi$	Cosine of angles ϕ, θ and ψ
$\mathbf{R}_{\mathrm{X}}^{\mathrm{Y}}$	Rotation matrix of a vector represented in an arbitrary *frame* X for an arbitrary Y *frame*
p	Angular speed related to the x axis in the *Frame* **B**
q	Angular speed related to the y axis in the *Frame* **B**
r	Angular speed related to the z axis in the *Frame* **B**

$\dot{p}$	Angular acceleration related to the x axis in the *Frame* **B**
$\dot{q}$	Angular acceleration related to the y axis in the *Frame* **B**
$\dot{r}$	Angular acceleration related to the z axis in the *Frame* **B**
η	Angular position vector in the *Frame* **I**
$\mathbf{r}$	Linear position vector in the *Frame* **I**
$\ddot{\mathbf{r}}$	Linear acceleration vector in the *Frame* **I**
ν	Angular speed vector in the *Frame* **B**
$\mathbf{T}$	Rotation matrix for angular velocity
L	Size of the quadrotor arm
$\mathbf{J}$	Inertia matrix
x	Linear position of quadrotor in the x axis of *frame* I
y	Linear position of quadrotor in the y axis of *frame* I
z	Linear position of quadrotor in the z axis of *frame* I
$\mathbf{e_r}$	Linear position error vector
$\mathbf{e_r}$	Linear velocity error vector
K_p	Linear control gain
K_d	Derivative control gain
m	Quadrotor mass
g	Gravity constant
K_R	Angular position control gain
K_ν	Angular velocity control gain
$\mathbf{t}$	Desired orientation vector
$\mathbf{b_i}$	Desired direction vector on axis i of *frame* B
$\mathbf{I_{3x3}}$	Identity matrix
$\mathbf{v}$	Axis of rotation of Rodriguez formula
β	Angle of rotation of Rodriguez formula
$\mathbf{e_R}$	Angular position error vector
$\mathbf{e}_\nu$	Angular velocity error vector
Ψ	Basin of attraction
α	Particle of the PSO algorithm
k	Constant of the rotor force
b	Constant of the rotor moment
τ	Time constant for rotor delay dynamics

Author details

Manuel A. Rendón[1,2]

1 Federal University of Juiz de Fora - UFJF, Juiz de Fora, MG, Brazil

2 Electromechanical Energy Conversion Group - GCEME, Juiz de Fora, Brazil

*Address all correspondence to: manuel.rendon@ufjf.edu.br

IntechOpen

© 2021 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

References

[1] Annaz F. Uav testbed training platform development using panda3d. Industrial Robot: An International Journal. 2015;**42**(5):450-456

[2] A Zul Azfar and D Hazry. Simple gui design for monitoring of a remotely operated quadrotor unmanned aerial vehicle (uav). In *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, pages 23–27. IEEE, 2011.

[3] A Tayebi and S McGilvray. Attitude stabilization of a four-rotor aerial robot. In Decision and Control, 2004. CDC. *43rd IEEE Conference on*, volume 2, pages 1216–1221. IEEE, 2004.

[4] Tiago P Nascimento and Martin Saska. Position and attitude control of multi-rotor aerial vehicles: A survey. *Annual Reviews in Control*, 2019.

[5] Can Dikmen I, Arisoy A, Hakan Temeltas. Attitude control of a quadrotor. In Recent Advances in Space Technologies. *RAST'09*. In: *4th International Conference on*, pages 722–727. IEEE. *2009*. p. 2009

[6] Gabriel M Hoffmann, Haomiao Huang, Steven L Waslander, and Claire J Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proc. of the AIAA Guidance, Navigation, and Control Conference*, volume 2, page 4, 2007.

[7] Domingues JMB. Quadrotor prototype. *Dissertacio*: *Uneversidade Tecnica deLisboa*; 2009

[8] Ly Dat Minh and Cheolkeun Ha. Modeling and control of quadrotor mav using vision-based measurement. In *Strategic Technology (IFOST), 2010 International Forum on*, pages 70–75. IEEE, 2010.

[9] Samir Bouabdallah and Roland Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2247–2252. IEEE, 2005. ISBN 078038914X.

[10] C Nicol, CJB Macnab, and A Ramirez-Serrano. Robust neural network control of a quadrotor helicopter. In *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*, pages 001233–001238. IEEE, 2008.

[11] Steven Lake Waslander, Gabriel M Hoffmann, Jung Soon Jang, and Claire J Tomlin. Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3712–3717. IEEE, 2005.

[12] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. Rotors—a modular gazebo mav simulator framework. In *Robot Operating System (ROS)*, pages 595–625. Springer, 2016.

[13] Subhan Khan, Mujtaba Hussain Jaffery, Athar Hanif, and Muhammad Rizwan Asif. Teaching tool for a control systems laboratory using a quadrotor as a plant in matlab. *IEEE Transactions on Education*, 60(4): 249–256, 2017.

[14] Manuel A Rendón and Felipe F Martins. Unmanned quadrotor path following nonlinear control tuning using particle swarm optimization. In *2018 Latin American Robotic Symposium, LARC 2018 and 2018 Workshop on Robotics in Education WRE 2018*, pages 509–514. IEEE, 2018.

[15] Manuel A Rendón, Felipe F Martins, and Luis Gustavo Ganimi. A visual interface tool for development of quadrotor control strategies. *Journal of Intelligent & Robotic Systems*, pages 1–18, 2020.

[16] Mellinger D, Michael N, Kumar V. Trajectory generation and control for precise aggressive maneuvers with quadrotors. The International Journal of Robotics Research. 2012;**31**(5):664-674

[17] Lee T, Leoky M, N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In Decision and Control (CDC). *49th IEEE Conference on*, pages 5420–5425. IEEE. *2010*;**2010**

[18] Randal Beard. Quadrotor dynamics and control rev 0.1. Technical document, Brigham Young University, Provo, UT, USA, February 2008.

[19] 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.

[20] Jian S Dai. Euler–Rodrigues formula variations, quaternion conjugation and intrinsic connections. Mechanism and Machine Theory, 92: 144–152, 2015.

[21] Felipe F. Martins and Manuel A. Rendón. Easyquadsim: Easy quadrotor simulator. GitHub, 2019. URL EasyQuadSim.