Simulation of a Fully Actuated Quadrotor UAV with a Propeller Tilting Mechanism

Antonio Bozza 1836119
Saverio Taliani 1841465
Valentina Becchetti 1739063

Supervisor:
Marilena Vendittelli

Abstract

The quadrotor unmanned aerial vehicles (UAVs) are a type of underactuated system that come equipped with four actuators. These UAVs have a strong coupling between their lateral motion and body orientation, which makes it challenging to track an arbitrary 6D trajectory in space. In the following report, a new quadrotor design has been analyzed that uses two additional actuators to control the tilt angles of the propellers relative to the quadrotor body.

The dynamic model derived for this new quadrotor design does not directly include the velocity of the controlled tilt angles of the propellers. Therefore, the system cannot be static feedback linearized. However, it can be linearized at a higher differential order, resulting in a dynamic feedback linearization controller. The system is visualized and simulated both in Matlab and in the CoppeliaSim environment, and the results of specific tasks are shown, demonstrating the effectiveness of the additional actuations, compared to the classical solution.

Contents

1 Introduction

Multirotor unmanned aerial vehicles (UAVs) have become increasingly popular due to their versatility and ease of use. These vehicles, equipped with multiple rotors and a central controller, are able to perform a huge variety of tasks. The most common types of multirotor UAVs are quadcopters, hexacopters, and octocopters, which feature four, six, and eight rotors, respectively.

Thanks to the possibility to take high quality videos from interesting perspectives, to their velocity and their agility and possibility to access impervious areas, drones suit multiple applications in various industries, such as videography, mapping, search and rescue operations, inspection of infrastructure, agricultural monitoring, and delivery services.

However, UAVs also have limitations due to their underactuation, given by the fact that the number of inputs is fewer than the number of degrees of freedom that need to be controlled. For example, a quadcopter has four rotors but needs to control six degrees of freedom given by the rigid body position and orientation in space. This can make it challenging to achieve certain maneuvers and trajectories.

Despite their limitations, these vehicles have shown great potential in a wide range of applications, as mentioned above. Ongoing research in control and optimization techniques, will continue to extend the abilities of these aircraft and further enhance their utility in various industry fields.

The following discussion propose: the improvement of the characteristics that make today's UAV perfect for the applications presented above. In particular it is analyzed the possibility to make a quadrotor more maneuverable, adding two actuation that together with the four already present makes it a fully actuated system.

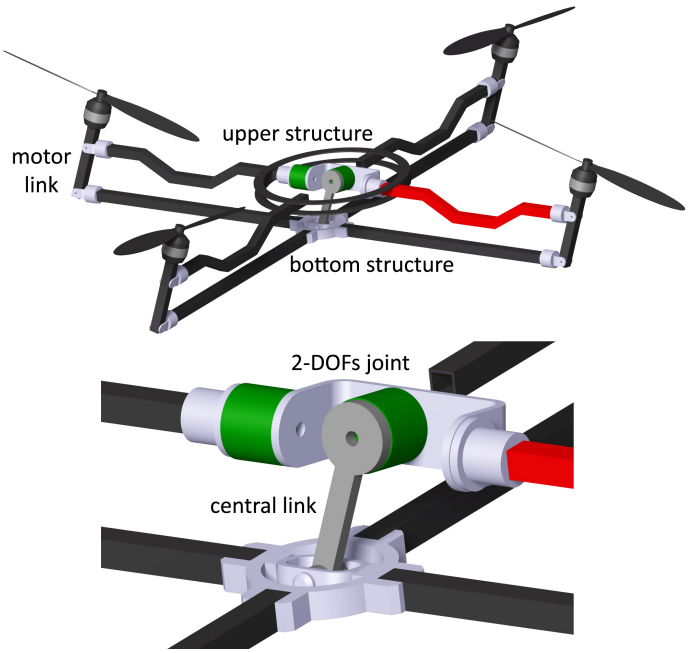


Figure 1: Top: conceptual CAD model of the proposed platform; bottom: the tilting mechanism in detail, in green the two additional actuators, in gray the joints of the platform

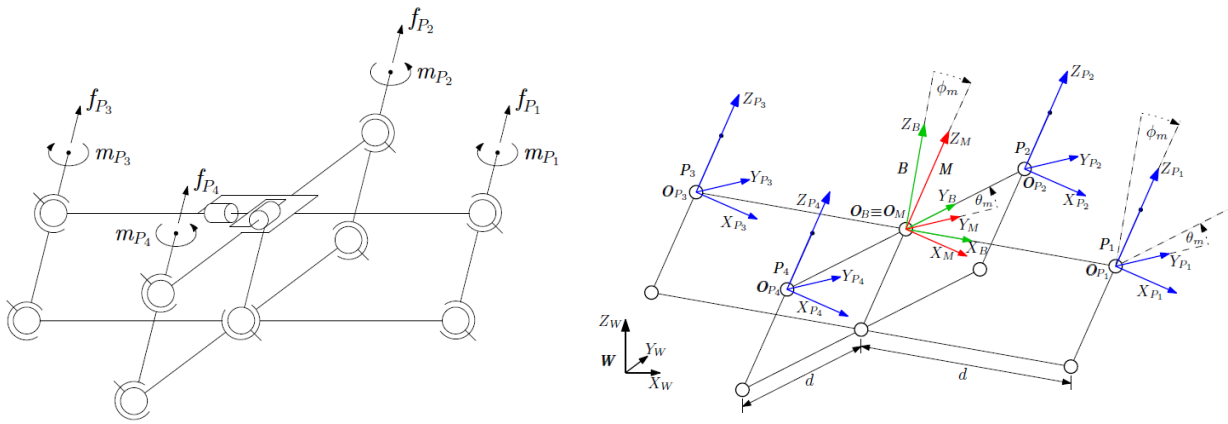
In [1] it is proposed an alternative actuation scheme, including a tilting mechanism for the UAV propellers, that makes it possible to regulate the roll and pitch angles, which is not possible for a classical UAV quadrotor structure, and gives the possibility to move in any direction independently.

As a matter of fact, the obtained full actuation makes it simpler for these vehicles to succeed in some applications that need precision in the movements and positioning tasks, in particular in photography and videography.

2 Platform description and Dynamic Model

In order to execute a pre-defined trajectory using a conventional quadrotor, it is imperative to ensure that the accumulated thrust from all four propellers is directed towards the intended direction while simultaneously counterbalancing any external disturbance or gravitational forces [2]. Despite the fact that the quadrotor's rotation can be induced through the resulting torque, the independent control of only one of the three angles of rotation (i.e., around the vertical axis or Z-axis) is possible while the other two are employed to orient the thrust in the aforementioned manner.

To address this challenge and regain control over the two degrees of freedom that are missing due to the coupling of the lateral motion with the tilt angles of the platform, in [1] a model featuring a propeller tilting mechanism is presented. This mechanism is capable of altering the orientation of the thrust generated by the propellers, thus allowing us to decouple the lateral motion from the tilt angles of the platform. This tilting mechanism, as illustrated in Figures 1 and 2, comprises a 2-degree-of-freedom (2-DOFs) actuated joint, a central link, and the bottom structure (consisting of the four bottom arms shown in the figures), which transfers the motion to the motor links. The central link, bottom structure, platform's arms, and motor links are interlinked to form four parallelograms coupled with the central link, enabling the simultaneous tilting of all the propellers.



(a) (b)

Figure 2: (a) Kinematic diagram of the platform and (b) Simplified diagram of the platform with depicted coordinate frames: W in black, B in green, M in red and P i, i = 1, . . . , 4 in blue..

In the process of deriving the dynamic model of this platform, which will be presented in this section, it is assumed that the motion of the bottom structure is negligible

in relation to the dynamics of the UAV. This assumption is motivated by the fact that the vast majority of the system's components, including the battery, computational unit, and others, are located around the center of the upper structure, and hence their mass is concentrated in this region.

2.1 Structure

In order to represent all the elements and their movement the following reference frame are defined: a global inertial frame of reference $\boldsymbol{W} : \{\boldsymbol{O}_W, X_W, Y_W, Z_W\}$; a moving body frame $\boldsymbol{B} : \{\boldsymbol{O}_B, X_B, Y_B, Z_B\}$ attached to the quadrotor at its center of mass, namely the center of the body equidistant from the propellers; the tilting mechanism frame $\boldsymbol{M} : \{\boldsymbol{O}_M, X_M, Y_M, Z_M\}$ with $\boldsymbol{O}_M \equiv \boldsymbol{O}_B$; and a set of four frames attached to the centers of propellers $\boldsymbol{P}_i : \{\boldsymbol{O}_{P_i}, X_{P_i}, Y_{P_i}, Z_{P_i}\}, i = 1 \dots 4$.

We can write the relative position and orientation of two frames $\boldsymbol{A}$ and $\boldsymbol{B}$ as ${}^B\boldsymbol{p}_A = {}^B\boldsymbol{O}_A \in R^3$ for position and ${}^B\boldsymbol{R}_A \in SO(3)$ for orientation.
Hence the body frame will have the sequent coordinates in world frame W:

$$\begin{aligned} \boldsymbol{p} &= {}^W\boldsymbol{p}_B = (p_x p_y p_z)^T \\ \boldsymbol{R}_B &= {}^W\boldsymbol{R}_B = \boldsymbol{R}_z\left({}^W\psi_B\right)\boldsymbol{R}_y\left({}^W\theta_B\right)\boldsymbol{R}_x\left({}^W\phi_B\right) \end{aligned} \tag{1}$$

where ${}^W\phi_B = \phi_B, {}^W\theta_B = \theta_B, {}^W\psi_B = \psi_B$ denote the roll, pitch and yaw rotation angles of the platform, and $\boldsymbol{R}_x(\cdot), \boldsymbol{R}_y(\cdot), \boldsymbol{R}_z(\cdot)$, the canonical rotation matrices representing the elemental rotations around the X, Y and Z axes.

The new design allows for independent application of force along a new direction Z_M, achieved by reorienting the propellers using a tilting mechanism, unlike in a classical quadrotor where the thrust is solely directed along the vertical axis Z_B of the body frame and is affected by body rotation.
As a consequence propellers and trust are tilted in the following way:

$${}^B\boldsymbol{R}_M = \boldsymbol{R}_y\left(\theta_m\right)\boldsymbol{R}_x\left(\phi_m\right) \tag{2}$$

where ${}^B\phi_M = \phi_m$ and ${}^B\theta_M = \theta_m$ are the roll and pitch angles of the tilting mechanism. The advantage of the parallelogram structure can be seen in the final orientation of the propellers where ${}^B\boldsymbol{R}_{P_i}$ is identical and equal to the orientation of the tilting mechanism

$${}^B\boldsymbol{R}_{P_i} = {}^B\boldsymbol{R}_M, \quad i = 1 \dots 4, \tag{3}$$

while their positions are simply defined as

$$\boldsymbol{O}_{P_i} = {}^{B}\boldsymbol{O}_{P_i} = \boldsymbol{R}_z((i-1)\pi/2) \begin{bmatrix} d \\ 0 \\ 0 \end{bmatrix}, \quad i = 1 \dots 4, \tag{4}$$

where d is the arm length - the distance from the centers of the propellers to the center of mass of the UAV.

$$\boldsymbol{F}_m = \begin{bmatrix} k_f c_{\phi_m} s_{\theta_m} & k_f c_{\phi_m} s_{\theta_m} & k_f c_{\phi_m} s_{\theta_m} & k_f c_{\phi_m} s_{\theta_m} \\ -k_f s_{\phi_m} & -k_f s_{\phi_m} & -k_f s_{\phi_m} & -k_f s_{\phi_m} \\ k_f c_{\phi_m} c_{\theta_m} & k_f c_{\phi_m} c_{\theta_m} & k_f c_{\phi_m} c_{\theta_m} & k_f c_{\phi_m} c_{\theta_m} \end{bmatrix} \tag{5}$$

$$\boldsymbol{\tau}_m = \begin{bmatrix} -k_m c_{\theta_m} s_{\phi_m} & dk_f c_{\theta_m} c_{\phi_m} + k_m c_{\theta_m} s_{\phi_m} & -k_m c_{\theta_m} s_{\phi_m} & -dk_f c_{\theta_m} c_{\phi_m} + k_m c_{\theta_m} s_{\phi_m} \\ -dk_f c_{\phi_m} c_{\theta_m} + k_m s_{\phi_m} & -k_m s_{\phi_m} & dk_f c_{\phi_m} c_{\theta_m} + k_m s_{\phi_m} & -k_m s_{\phi_m} \\ -dk_f s_{\phi_m} - k_m c_{\phi_m} c_{\theta_m} & -dk_f c_{\phi_m} s_{\theta_m} + k_m c_{\phi_m} c_{\theta_m} & dk_f s_{\phi_m} - k_m c_{\phi_m} c_{\theta_m} & dk_f c_{\phi_m} s_{\theta_m} + k_m c_{\phi_m} c_{\theta_m} \end{bmatrix}$$

2.2 Dynamic model

To simulate the thrust the following mathematical formulation is considered:

$$\begin{cases} \boldsymbol{f}_{P_i} = (0 \ 0 \ k_f \bar{w}_i |\bar{w}_i|)^T, & k_f > 0, \\ \boldsymbol{m}_{P_i} = (0 \ 0 \ -k_m \bar{w}_i |\bar{w}_i|)^T, & k_m > 0, \end{cases}$$

in which the produced thrust $\boldsymbol{f}_{P_i}$ and the reaction moment $\boldsymbol{m}_{P_i}$ in the propeller frame $\boldsymbol{P}_i, i = 1 \dots 4$ are proportional to the signed square spinning velocity of the rotor $w_i = \bar{w}_i |\bar{w}_i|$, $i = 1 \dots 4$ with factors k_f and $-k_m$, respectively. The factors k_f and k_m are thrust and torque propeller's coefficients. Hence, we can obtain the total propellers thrust ${}^{B}\boldsymbol{T} \in R^3$ and torque ${}^{B}\boldsymbol{\tau} \in R^3$ acting on the center of mass

$$\begin{aligned} {}^{B}\boldsymbol{T} &= \sum_{i=1}^{4} {}^{B}\boldsymbol{R}_{P_i} \boldsymbol{f}_{P_i}, \\ {}^{B}\boldsymbol{\tau} &= \sum_{i=1}^{4} {}^{B}\boldsymbol{R}_{P_i} \boldsymbol{m}_{P_i} + \sum_{i=1}^{4} \left({}^{B}\boldsymbol{O}_{P_i} \times {}^{B}\boldsymbol{R}_{P_i} \boldsymbol{f}_{P_i} \right), \end{aligned}$$

Substituting (3) and (4) to the Newton-Euler set of equations

$$\begin{aligned} \boldsymbol{R}_B^B \boldsymbol{T} &= m \left(\ddot{\boldsymbol{p}} - \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \right), \\ {}^{B}\boldsymbol{\tau} &= \boldsymbol{I}_B \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times \boldsymbol{I}_B \boldsymbol{\omega}_B, \end{aligned}$$

yields the dynamic model of our system:

$$\begin{aligned}\begin{bmatrix} \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{\omega}}_B \end{bmatrix} &= \begin{bmatrix} \boldsymbol{g} \\ \mathbf{c} \end{bmatrix} + \begin{bmatrix} \frac{1}{m}\boldsymbol{R}_B & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_B^{-1} \end{bmatrix} \begin{bmatrix} \frac{\partial^B \boldsymbol{T}}{\partial \boldsymbol{w}} & \frac{\partial^B \boldsymbol{T}}{\partial \boldsymbol{\omega}_m} \\ \frac{\partial^B \boldsymbol{\tau}}{\partial \boldsymbol{w}} & \frac{\partial^B \boldsymbol{\tau}}{\partial \boldsymbol{\omega}_m} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\omega}_m \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{g} \\ \mathbf{c} \end{bmatrix} + \begin{bmatrix} \frac{1}{m}\boldsymbol{R}_B & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_B^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{F}_m & \mathbf{0} \\ \boldsymbol{\tau}_m & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\omega}_m \end{bmatrix} \\ &= \boldsymbol{f} + \boldsymbol{J}_R \begin{bmatrix} \boldsymbol{J}_m & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\omega}_m \end{bmatrix} = \boldsymbol{f} + \boldsymbol{J} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\omega}_m \end{bmatrix}\end{aligned} \tag{6}$$

where m is the total mass of the platform, $\boldsymbol{I}_B \in R^{3\times 3}$ is the constant, diagonal and positive-definite inertia matrix, $\boldsymbol{\omega}_B = \begin{pmatrix}\omega_{B_x}\omega_{B_y}\omega_{B_z}\end{pmatrix}^T$ is the angular velocity of the UAV in $\boldsymbol{B}$,

$$\boldsymbol{f} = \begin{bmatrix}\boldsymbol{g}^T \boldsymbol{c}^T\end{bmatrix}^T = \left[\begin{pmatrix} 0 & 0 & -g \end{pmatrix}^T \left(-\boldsymbol{I}_B^{-1}\left(\boldsymbol{\omega}_B \times \boldsymbol{I}_B \boldsymbol{\omega}_B\right)\right)^T \right]^T \tag{7}$$

expresses the gravity force acting on the system and the Coriolis term,

$$\begin{bmatrix}\boldsymbol{w}^T \boldsymbol{\omega}_m^T\end{bmatrix}^T = \left[\begin{pmatrix} w_1 & w_2 & w_3 & w_4 \end{pmatrix}^T \left(\dot{\phi}_m \dot{\theta}_m\right)^T \right]^T \tag{8}$$

is the control input of the system, and $\boldsymbol{F}_m = \frac{\partial^B \boldsymbol{T}}{\partial \boldsymbol{w}}, \boldsymbol{\tau}_m = \frac{\partial^B \tau}{\partial \boldsymbol{w}}$ are given by equations (8) and (9), respectively ($c_\alpha = \cos(\alpha)$ and $s_\alpha = \sin(\alpha)$ for $\alpha = \phi_m, \theta_m$).

3 Control Design and Implementation

In order to validate and to further test the structure and the control proposed, we implemented both the model and the controller in MATLAB, simulating its behaviour in some relevant tasks for the new structure proposed.

A similar construction is implemented also in CoppeliaSim environment to visualize the control effects. To model in a simple way the propeller tilting mechanism, the body is constituted by a simple cross structure to which the propellers are attached through two revolute joints. In this setting the joints are considered perfect actuators, reaching a commanded position instantaneously, all at the same time, fully substituting the parallelogram structure. This simplification allows to clearly visualize and regulate propellers orientation.

3.1 Control implementation

The drone's control in the CoppeliaSim scene is based on applying the thrust and torque at each propeller. Each thrust is applied to the center of each propeller responsible, and through the two revolute joints, it can be rotated according to the control of the new DoF of this structure. As a matter of fact, in the script associated with the drone shape it is implemented a step of the control loop described in [1], generating the w_i and the $\dot{\phi}_m, \dot{\theta}_m$ to decide on thrust and torque, according to the formulation above.

Being not possible to statically feedback linearize the model in (6), it is considered at an higher differential order, and the final control is computed inverting the following derivation:

$$\begin{aligned}
\begin{bmatrix} \ddot{\boldsymbol{p}} \\ \ddot{\boldsymbol{\omega}}_B \end{bmatrix} &= \dot{\boldsymbol{f}} + \dot{\boldsymbol{J}}_R \boldsymbol{J}_m \boldsymbol{w} + \boldsymbol{J}_R \dot{\boldsymbol{J}}_m \boldsymbol{w} + \boldsymbol{J}_R \boldsymbol{J}_m \dot{\boldsymbol{w}} \\
&= \begin{bmatrix} \frac{1}{m} \dot{\boldsymbol{R}}_B \boldsymbol{F}_m \boldsymbol{w} \\ \dot{\boldsymbol{c}} \end{bmatrix} + \boldsymbol{J}_R \begin{bmatrix} \frac{\partial \boldsymbol{F}_m}{\partial \phi_m} \boldsymbol{w} & \frac{\partial \boldsymbol{F}_m}{\partial \theta_m} \boldsymbol{w} \\ \frac{\partial \boldsymbol{\tau}_m}{\partial \phi_m} \boldsymbol{w} & \frac{\partial \tau_m}{\partial \theta_m} \boldsymbol{w} \end{bmatrix} + \boldsymbol{J}_R \boldsymbol{J}_m \dot{\boldsymbol{w}} \\
&= \boldsymbol{J}_R \begin{bmatrix} \boldsymbol{F}_m & \frac{\partial \boldsymbol{F}_m}{\partial \phi_m} \boldsymbol{w} & \frac{\partial \boldsymbol{F}_m}{\partial \theta_m} \boldsymbol{w} \\ \boldsymbol{\tau}_m & \frac{\partial \tau_m}{\partial \phi_m} \boldsymbol{w} & \frac{\partial \boldsymbol{\tau}_m}{\partial \theta_m} \boldsymbol{w} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{w}} \\ \boldsymbol{\omega}_m \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \dot{\boldsymbol{R}}_B \boldsymbol{F}_m \boldsymbol{w} \\ \dot{\boldsymbol{c}} \end{bmatrix} \\
&= \boldsymbol{J}^\star \begin{bmatrix} \dot{\boldsymbol{w}} \\ \boldsymbol{\omega}_m \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \dot{\boldsymbol{R}}_B \boldsymbol{F}_m \boldsymbol{w} \\ \dot{\boldsymbol{c}} \end{bmatrix},
\end{aligned} \tag{9}$$

where $\boldsymbol{J}^\star$ is the extended Jacobian, $\boldsymbol{w}$ becomes an internal state of the system. Following the above considerations, the script is strucrured following these steps:

1. current position, orientation, both linear and angular velocities are taken from the scene, through the functions *sim.getObjectPosition*, *sim.getObjectOrientation*

sim.getVelocity, the tilting angles of the propeller are taken from the angular position of the joints, the angular velocities of the propellers are stored from the previous step;

2. the main components of the control are computed as in (5) and (6): $\boldsymbol{R}_B$, $\boldsymbol{F}_m$, $\boldsymbol{\tau}_m$ and their analytic derivatives, but also $\boldsymbol{J}_m$ and

$$\boldsymbol{J}^* = \boldsymbol{J}_R \begin{bmatrix} \boldsymbol{F}_m & \frac{\partial \boldsymbol{F}_m}{\partial \phi_m}\boldsymbol{w} & \frac{\partial \boldsymbol{F}_m}{\partial \theta_m}\boldsymbol{w} \\ \boldsymbol{\tau}_m & \frac{\partial \tau_m}{\partial \phi_m}\boldsymbol{w} & \frac{\partial \boldsymbol{\tau}_m}{\partial \theta_m}\boldsymbol{w} \end{bmatrix} \tag{10}$$

3. the numerical derivative of the Coriolis term c is computed with a first order approximation;

4. the system (6) is feedback linearized with a reference input $\left[\ddot{\boldsymbol{p}}_r^T \dot{\boldsymbol{\omega}}_r^T\right]^T$ by means of the law:

$$\begin{bmatrix} \dot{\boldsymbol{w}} \\ \boldsymbol{\omega}_m \end{bmatrix} = \boldsymbol{J}^{\star -1} \left(\begin{bmatrix} \ddot{\boldsymbol{p}}_r \\ \dot{\boldsymbol{\omega}}_r \end{bmatrix} - \begin{bmatrix} \frac{1}{m}\dot{\boldsymbol{R}}_B \boldsymbol{F}_m \boldsymbol{w} \\ \dot{\boldsymbol{c}} \end{bmatrix} \right) \tag{11}$$

which has a solution if $\rho_{J^\star} = \operatorname{rank}(\boldsymbol{J}^\star) = 6$, e.i.,

$$\det(\boldsymbol{J}^\star) = 8 d^2 k_f^5 k_m c_{\phi_m}^2 c_{\theta_m} \cdot (w_1 + w_2 + w_3 + w_4)^2 \neq 0 \tag{12}$$

condition (12) is satisfied for $\phi_m, \theta_m \neq \pm\frac{\pi}{2}$ and $w_i > 0, i = 1 \dots 4$, which is always fulfilled due to the mechanical constrains of the platform $\left(|\phi_m|, |\theta_m| < \frac{\pi}{6}\right)$ and the assumption that during flight $w_i > 0, i = 1 \dots 4$.
In this setting it is employed a linear trajectory tracking controller for position:

$$\ddot{\boldsymbol{p}}_r = \ddot{\boldsymbol{p}}_d + \boldsymbol{K}_{p_1}(\ddot{\boldsymbol{p}}_d - \ddot{\boldsymbol{p}}) + \boldsymbol{K}_{p_2}(\dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}) + \boldsymbol{K}_{p_3}(\boldsymbol{p}_d - \boldsymbol{p}), \tag{13}$$

and one for the orientation

$$\ddot{\boldsymbol{\omega}}_r = \ddot{\boldsymbol{\omega}}_d + \boldsymbol{K}_{\omega_1}(\dot{\boldsymbol{\omega}}_d - \dot{\boldsymbol{\omega}}_B) + \boldsymbol{K}_{\omega_2}(\omega_d - \boldsymbol{\omega}_B) + \boldsymbol{K}_{\omega_3} e_R, \tag{14}$$

as a common approach for feedback linearized systems. The orientation error e_R is defined as

$$\boldsymbol{e}_R = \frac{1}{2}\left[\boldsymbol{R}_B^T \boldsymbol{R}_d - \boldsymbol{R}_d^T \boldsymbol{R}_B\right]_\vee \tag{15}$$

with $[\cdot]_\vee$ being the inverse map from $SO(3)$ to R^3.

For the linear trajectory tracking, although we can have analytically the derivatives of a smooth trajectory to follow, the acceleration at each step of the UAV in the CoppeliaSim environment is estimated numerically with a first order approximation from the linear and angular velocity data.

5. the input found at point 4 is integrated numerically, through an Euler integration, giving $\left[\boldsymbol{w}^T \ \phi_m \ \theta_m\right]^T$, through which we can impose the thrust at each propeller and tilt them commanding the two additional actuators.

The MATLAB implementation consists of three main scripts: *simulation.m*, *drone_system.m*, and *control_function.m*.

The *simulation.m* script initializes a struct called *drone*, which defines the state of the drone, but contains also the torque applied and the current propeller velocities. The script then iteratively calls the *control_function* and *drone_system* functions to simulate the evolution of the drone over time.

The *drone_system.m* script simulates a single step for the drone, taking as input the control action to perform and outputting the resulting drone struct.

The *control_function.m* script determines the appropriate control action for the drone based on the current state of the drone struct and the reference trajectory. The control action is then outputted.

The propeller accelerations obtained from *control_function.m* are integrated to obtain the control input for the system described by equation (6). This control input is denoted as $\left[\boldsymbol{w}^T \boldsymbol{\omega}_m^T\right]^T$.

3.2 Simulations results

In the simulations we considered the following table of values for some hyper-parameters that appears in the control above presented:

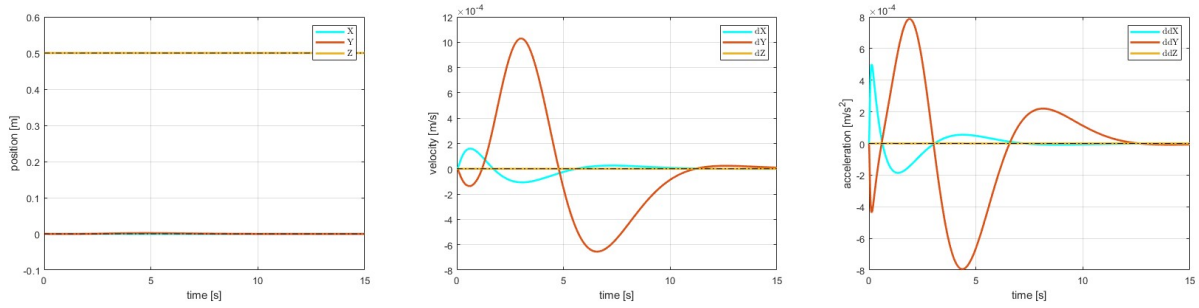
Parameter	Value
k_f	0.016
k_m	0.0027
K_{p1}	$24I_{3\times3}$
K_{p2}	$32I_{3\times3}$
K_{p3}	$16I_{3\times3}$
K_{om1}	$24I_{3\times3}$
K_{om2}	$32I_{3\times3}$
K_{om3}	$16I_{3\times3}$

Table 1: Main control parameters value

The quadcopter considered has a mass of $1\,kg$ and an arm lenght of $d = 0.3\,m$.
Three main simulations are presented below:

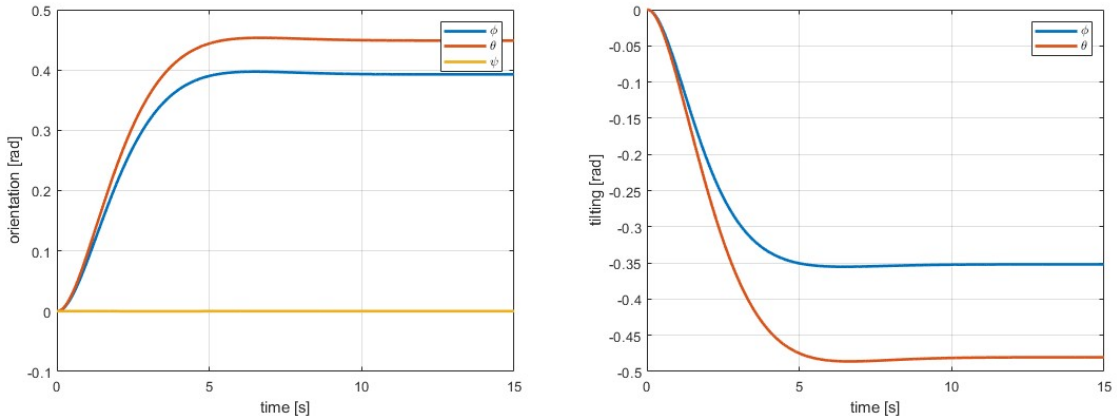
1. hovering task with non-zero orientation, to highlight the possibility to arbitrarily assign body orientation independently with respect to the drone Cartesian position;
2. linear lateral movement, to highlight the possibility to follow a linear path in the x-y plane without affecting the z component, thanks to the tilting mechanism;
3. tracking task for a spiral trajectory in the 3D space.

The hovering evolution is presented below:



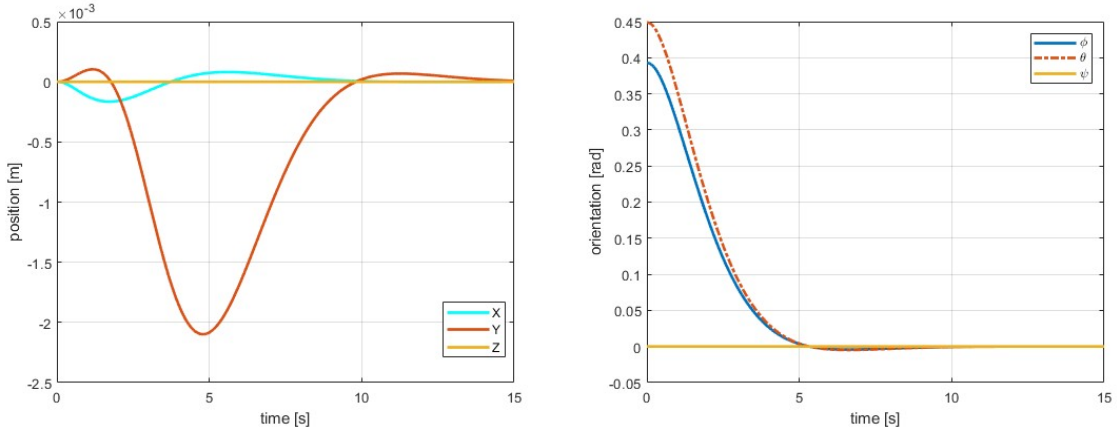
(a) (b) (c)

Figure 3: Cartesian error in position (a), velocities (b) and accelerations (c) of the hovering task



(a) (b)

Figure 4: Drone orientation (a) and propeller tilting angles (b) of the hovering task



(a) (b)

Figure 5: Cartesian position (a) and drone orientation (b) errors of the hovering task

As we can see from fig 3 and 4, the task is achieved moving only slightly, in the order of $10^{-4}m$ from the initial and final position $[0, 0, 0.5]$ during the evolution.

From 5 we can see the error plot both in position and in orientation for this task. The cartesian position is maintained precisely while the orientation of the body goes to its desired value, the small oscillations are rapidly recovered and have a negligible magnitude. The orientation error has its peak at the beginning due to the step input reference, and the target is reached with exponential convergence annihilating the error.

The final orientation of $[\pi/8, \pi/7, 0]$ is reached and the final tilting angles make the propeller thrust oriented in the z direction of the world frame to counteract gravity.

It is then presented a lateral motion task, in which the drone has to start from $[0, 0, 0.5]$, reach in sequence $[1, 1, 1]$, $[0, 2, 0.7]$ and return in the initial position, and we want to complete the task with a sequence of straight linear paths.

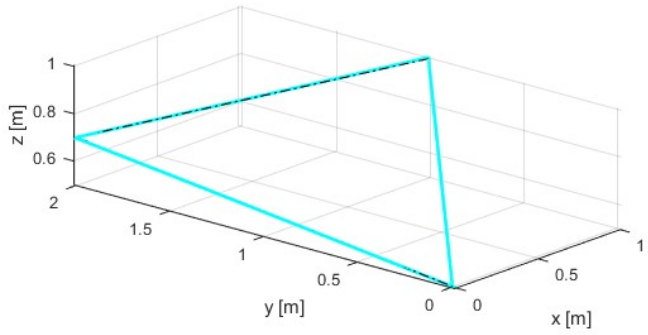
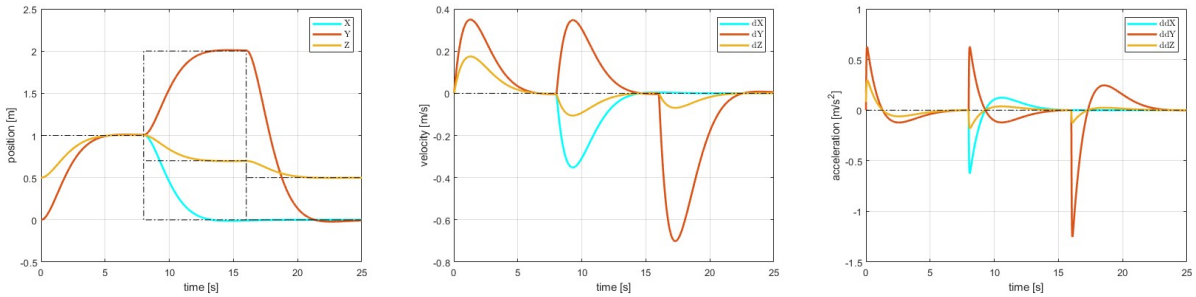


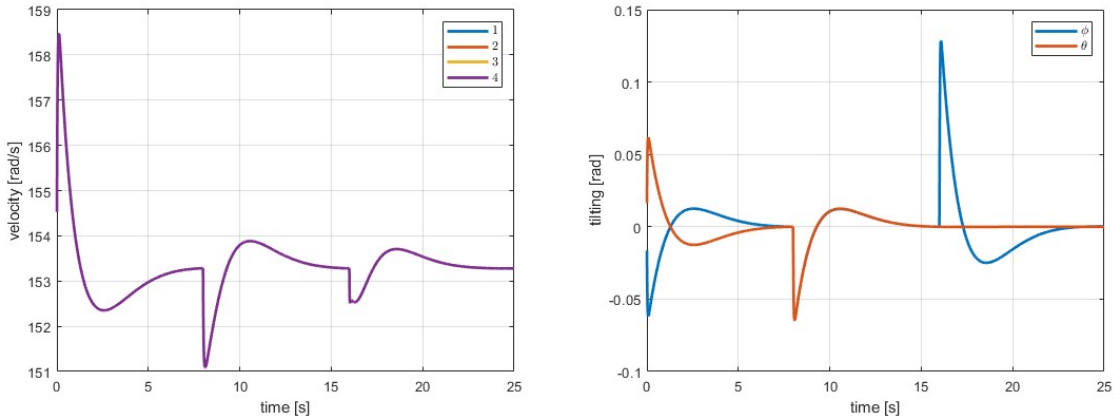
Figure 6: Linear 3D path



(a) (b) (c)

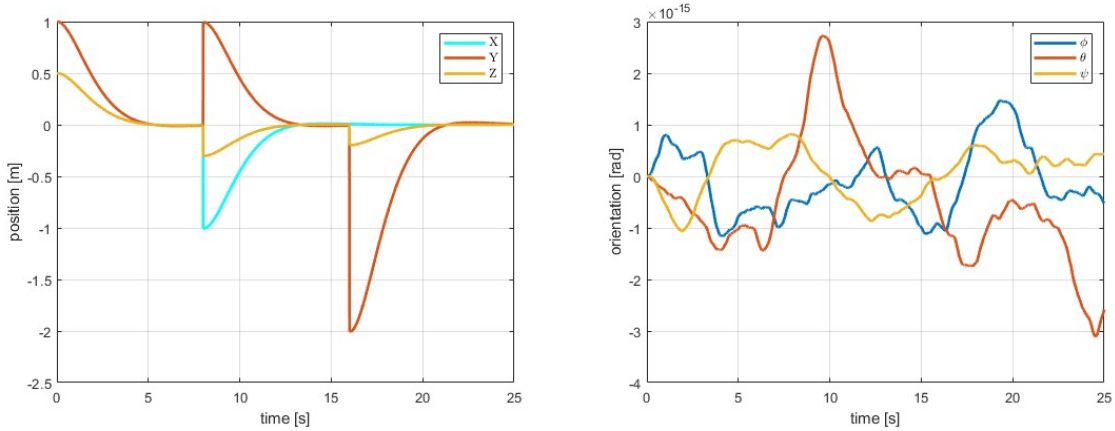
Figure 7: Cartesian position (a), velocities (b) and accelerations (c) of the linear path

In figures 7 are shown the cartesian position, velocity and acceleration evolution during this task, and we can see also from the executed 3D path in figure 6 that the sequence of goals is fully achieved. In figure 8 the spikes in the velocities of the propellers and the tilting angles, give us a hint of the goal switch and the direction in which the movement is directed.



(a) (b)

Figure 8: Propellers velocities (a) and tilting angles (b) of the linear path



(a) (b)

Figure 9: Cartesian position (a) and drone orientation (b) errors of the linear path

As a matter of fact, we have three spikes, that are linked to the change of the goal, both in the propeller rotation velocities and in their tilting angles. These changes allow to generate an acceleration (positive, increasing the propeller velocities, negative decreasing them), and to direct it (through propeller tilt).

In 9a the position error is shown, also in this case the reference is given as step input so the controller experiences a high error value that is successfully reduced each time a new reference is sent. While the quadrotor reaches the desired position as we can see from 9b the orientation is perfectly kept to its desired value, leaving the error below $3 \times 10^{-15} rad$.

Finally it is presented the spiral motion with the following reference time law:

$$x_r = cos(a)$$

$$y_r = sin(a)$$

$$z_r = 0.5 + a/10$$

The evolution, with the reference now different from zero for both acceleration and velocity, are shown in figure 10 and in 11. As we can see the controller makes the drone state coincide with the reference in less then five seconds, with a perfect tracking in time after this time value.

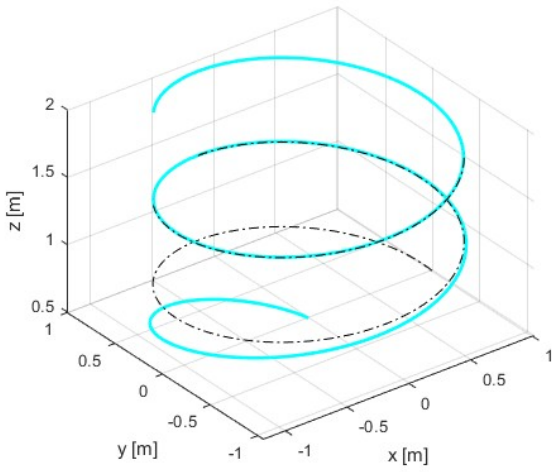
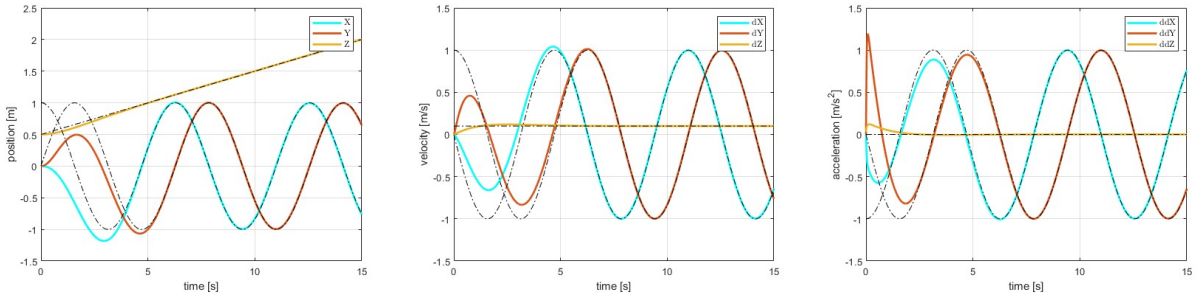
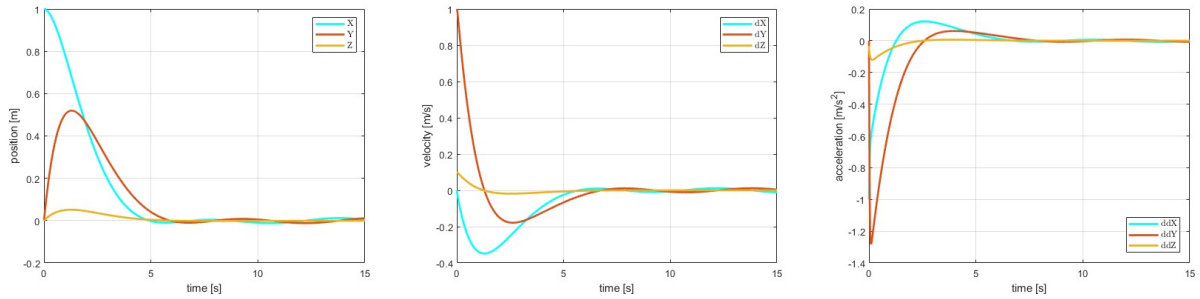


Figure 10: Tornado 3D path



(a) (b) (c)

Figure 11: Cartesian position (a), velocities (b) and accelerations (c) of the tornado path

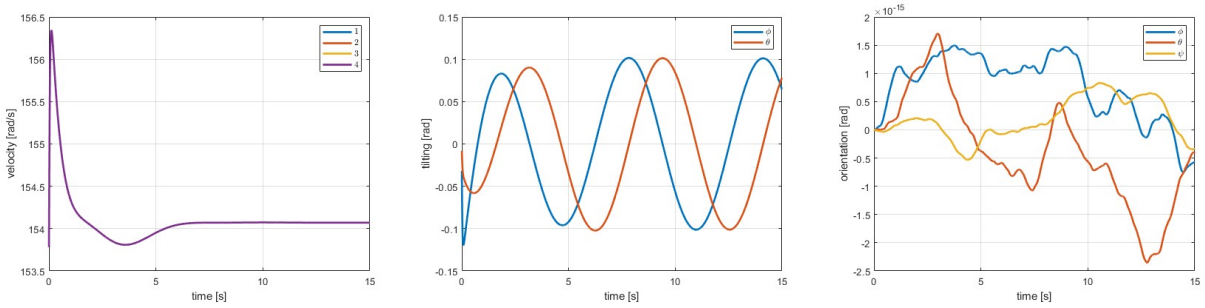


(a) (b) (c)

Figure 12: Cartesian position (a), velocities (b) and accelerations (c) errors of the tornado path

In figure 12 are presented the cartesian error in position, velocity and orientation with respect to the reference signal. As we can see after an initial transient the error reaches zero value for all the components of all three physical measures, leaving only a non-significant oscillation at steady-state, in the order of 10^{-2} for all the signals.

In figure 13a are also presented the propellers' velocities, that after the initial spike to reach the constant velocity to lift, stabilize around the value to compensate the gravity. On the other hand in figure 13b are presented the tilting angles, that follow the sinusoidal evolution that makes the orientation of the thrust align with the direction of the spiral while maintaining a constant orientation.



(a) (b) (c)

Figure 13: Propellers velocities (a), tilting angles (b) and drone orientation error of the tornado path

As a last comment while performing the spiral tracking, the orientation is subject to errors of the magnitude of 10^{-15} so we can consider it perfectly executed (13c).

4 Limitations and solutions

The controller proposed in the previous section can present some issues in particular cases. As a matter of fact, for mechanical reasons the tilting angles are limited, as presented also in the simulations in [1]. In our case we have limited the tilting angles to $\pi/6$ and this makes the controller unable to maintain stability during motions that require high lateral accelerations or hovering with the body tilted more than this value, with respect to the x or y axis.

The hovering problem cannot be solved by any means, being the tilting mechanism essential for this task (to maintain a vertical thrust) and making every alternative pointless while no other approach is able to stabilize the quadrotor in a fixed position with a desired body orientation. On the other hand, it is still possible to obtain better accelerations performance by deploying another controller.

Firstly, we investigated the possibility of limiting the Cartesian accelerations, by modifying the propeller's rotation speed. It is presented in figure 14 a movement that starts from [0,0,0.5] and finishes at [100,0,0.5]. In this case, the tilting angles are limited in module to $pi/6$, and with this specific trajectory, they saturate between seconds 2 and 3 and between seconds 6 and 7, as shown in 15a. Moreover, in figure 15b we can see that the orientation nearly does not change (the oscillations are in the order of 10^{-15}), as also seen in the previous simulations.

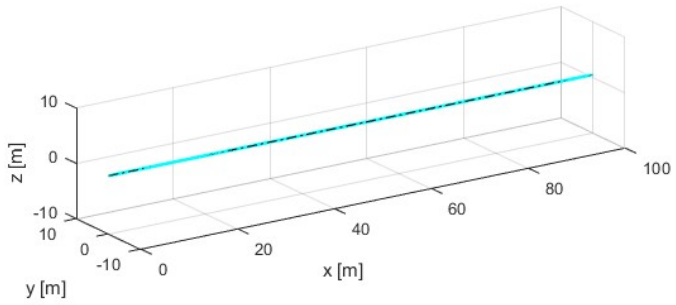
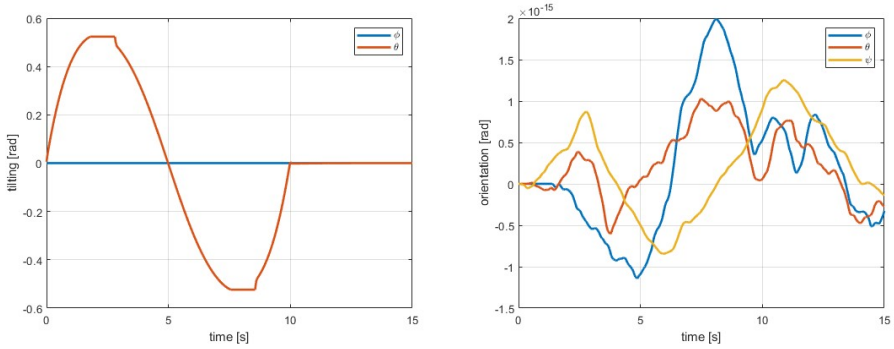


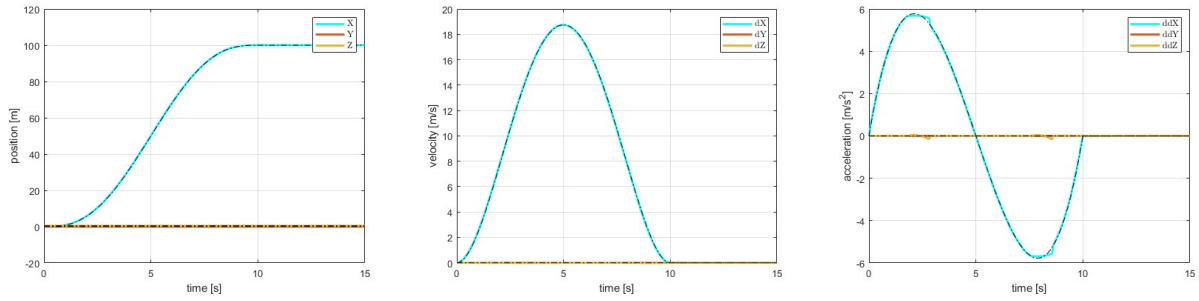
Figure 14: 3D path



(a) (b)

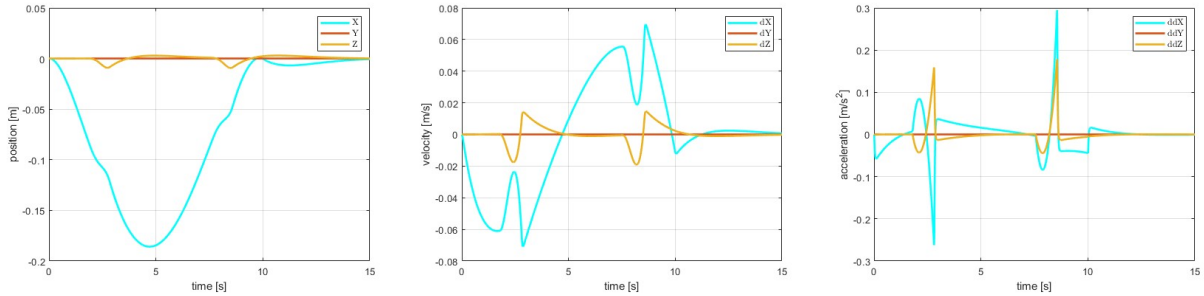
Figure 15: Propellers tilting angles (a) and body orientation error (b)

In figure 16 are presented the position, velocity and acceleration profiles, and as we can see the controller can still achieve a good tracking performance. On the other hand, if we analyze 17 we see that the X coordinate is followed rightly only in space, but produces a significant error for the time tracking due to the saturation.



(a) (b) (c)

Figure 16: Cartesian position (a), velocities (b) and accelerations (c)



(a) (b) (c)

Figure 17: Cartesian position (a), velocities (b) and accelerations (c) errors

This does not happen for an upscaled version in time of the same trajectory: in this case it is required a higher acceleration for a longer time period, so a tilting value higher than the maximum is required for a longer time period. This accumulates an error for the position and velocity tracking that makes the control input explode.

To overcome this issue we decided to employ a dynamic feedback linearization control suited for a conventional quadrotor without a tilting mechanism. When a tilting higher than $\pi/6$ is commanded we use the classical control to return to the presented control when the desired propeller angle is feasible by tilting only rotors.

4.1 Classic control implementation

Follows the implementation of the control of a classic quadcopter structure, through dynamic feedback linearization. In this second scenario, the drone's control is based on applying the thrust and torque at the centre of the full body, having all the propellers the same orientation.

The equations of motion of the quadrotor are:

$$\begin{aligned}
\ddot{x} &= u_1(\cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi) \\
\ddot{y} &= u_1(\sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi) \\
\ddot{z} &= u_1(\cos\phi \cos\theta) - g \\
\ddot{\phi} &= u_2 \\
\ddot{\theta} &= u_3 \\
\ddot{\psi} &= u_4
\end{aligned} \tag{16}$$

In this case the inputs are different from the previous model:

- u_1 is the thrust magnitude in the z coordinate of the UAV body frame;
- $[u_2, u_3, u_4]$ are the torques respectively with respect to the x, y, z UAV body frame.

We present a non-linear position controller for a quadrotor using feedback linearization with dynamic extension, the developed controller completely decouples and linearizes the nonlinear dynamical model of the aircraft [3].
The quadrotor output dynamics can be placed in state space form as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} c\psi s\theta c\phi + s\psi s\phi & 0 & 0 & 0 \\ s\psi s\theta c\phi - c\psi s\phi & 0 & 0 & 0 \\ c\phi c\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{17}$$

The matrix in (17) is singular so it means that there is no feedback that will linearize the dynamics. We must use dynamic inversion and this can be achieved by dynamic extension or simply by placing two integrator before the input u_1. Differentiating two more times $\ddot{x}$, $\ddot{y}$, $\ddot{z}$ from (16) we obtain the following output dynamics for the quadrotor:

$$\begin{bmatrix} x^{(4)} \\ y^{(4)} \\ z^{(4)} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ 0 \end{bmatrix} + \begin{bmatrix} g_{xu_1} & g_{xu_2} & g_{xu_3} & g_{xu_4} \\ g_{y\ddot{u}_1} & g_{yu_2} & g_{yu_3} & g_{yu_4} \\ g_{zu_1} & g_{zu_2} & g_{zu_3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{u}_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{18}$$

where $f^{,s}$ and $g^{,s}$ are nonlinear functions of state and their derivatives. The dynamic state feedback law that linearize and decouple the quadrotor outputs can be calculated as follows:

$$\begin{bmatrix} \ddot{u}_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} g_{x\ddot{u}_1} & g_{xu_2} & g_{xu_3} & g_{xu_4} \\ g_{y\ddot{u}_1} & g_{yu_2} & g_{yu_3} & g_{yu_4} \\ g_{z\ddot{u}_1} & g_{zu_2} & g_{zu_3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} v_1 - f_x \\ v_2 - f_y \\ v_3 - f_z \\ v_4 \end{bmatrix} \tag{19}$$

where v_1, v_2, v_3 and v_4 are new control inputs such that resulting closed loop system is in the form:

$$\begin{bmatrix} x^{(4)} \\ y^{(4)} \\ z^{(4)} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{20}$$

The control law (19) fully linearize the quadrotor dynamics described by (16) and leave no zero dynamics. The inverse of the decoupling matrix in (19) is non singular as along as u_1 is nonzero. This fact agrees well with the intuition that no amount of rolling or pitching will effect the motion of the quadrotor if there is no thrust to effect the acceleration [3].

4.2 Hybrid control implementation

The next step is to mix the two controls that we have presented in this report. To do so we use the tilting mechanism to execute each trajectory which requires the propellers to have an orientation smaller than $\pi/6$. When this condition is not met for example when a high lateral acceleration is required, in order to avoid the error to undermine the stability of the quadrotor, we switch to the classical controller sacrificing the body orientation but achieving higher performance.
To ensure continuity between the controllers we force the orientation of the UAV body to meet the propeller orientation after the switch. We consider, thanks to the tilting mechanism, that the propeller orientation with respect to the world frame is comparable to the body orientation of a traditional quadcopter.
In this way the following changes are made in switch instants:

- tilting controller $\rightarrow$ classic controller:
    1. the drone orientation is assumed to have the same values in the ϕ_x and θ_y coordinates as the propeller frame orientation;
    2. the thrust and torques are assumed to be realizable in this new setting (considering that the virtual drone has different distances between the propellers)
- classic controller $\rightarrow$ tilting controller: the drone orientation is returned to the real body orientation, subtracting the values of the propeller's tilt;

We switch back to the tilting mechanism control when the required body orientation, of the virtual drone is less than $pi/6$, assuring again the possibility to follow the desired trajectory and set a desired body orientation.

In MATLAB this hybrid setting is realized in the *hybrid_simulation.m* script, in which it is chosen at each simulation step the controller and the corresponding drone model to simulate, setting the described changes when a switch occurs. For this implementation, functions for the control and simulation of the classic drone structure are called in the script, respectively *classic_uav_control.m* and *classic_drone_system.m*, with the same scheme as the new drone functions described in section 3.1.

4.3 Simulations results

To check the effectiveness of the hybrid control above presented, it is considered the same trajectory seen in figure 14 but requiring a higher lateral acceleration which was not achievable before.

First, we consider the classic controller to check its behavior for the discussed trajectory.

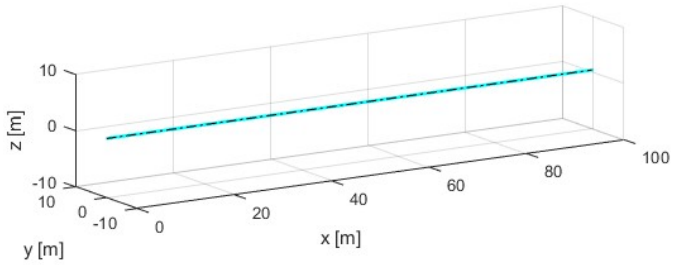
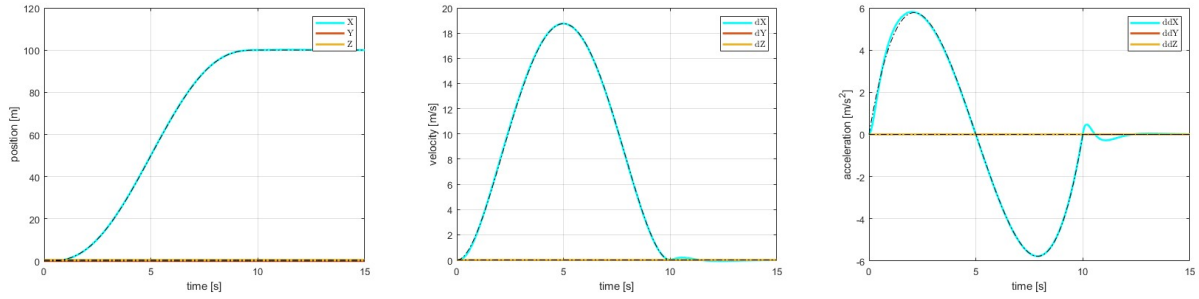


Figure 18: 3D path with the classic controller

As we can see from figures 18 and 19, the task is executed without problems, with the cartesian position, velocity and acceleration profile followed correctly. As expected the UAV orientation follows the acceleration profile, to give the right orientation to the thrust, as shown in figure 20.



(a) (b) (c)

Figure 19: Cartesian position (a), velocities (b) and accelerations (c) with the classic controller

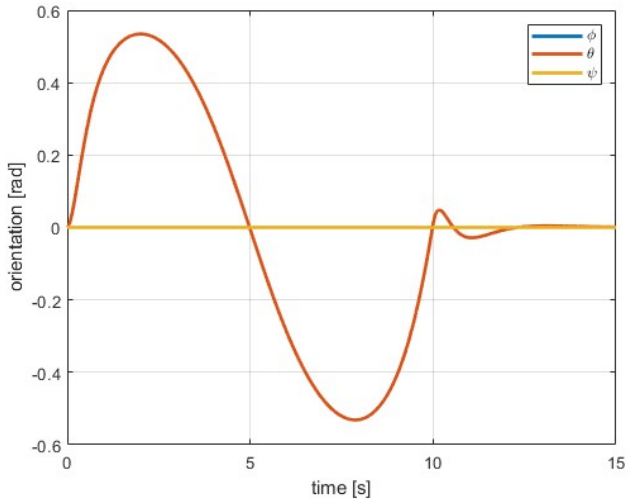
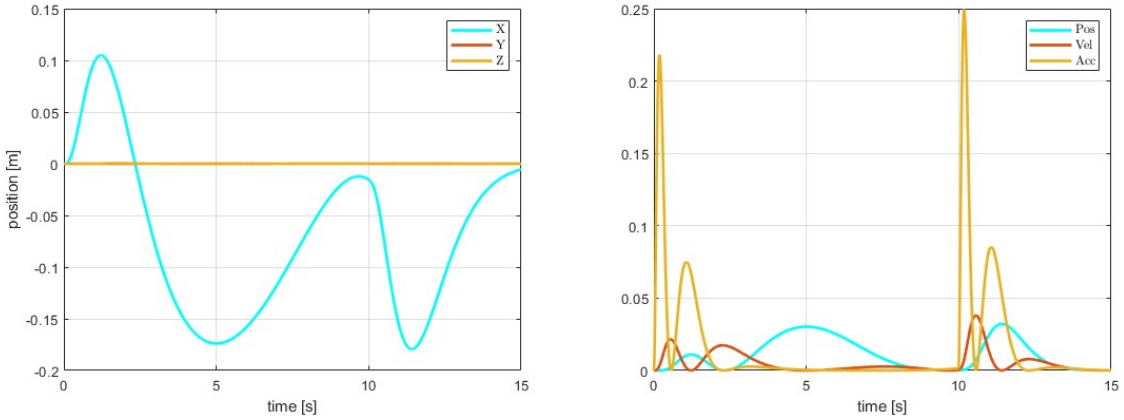


Figure 20: 3D path with the classic controller

In addition from figure 21 we can highlight the significant position errors in the trajectory tracking in time, and the quadratic norm of the error vectors for position, velocity and acceleration.



(a) (b)

Figure 21: Cartesian position (a) and quadratic norm (b) errors with the classic controller

For the same task, it is presented the result with the hybrid control scheme:
as we can see from figure 22, the path is followed perfectly, having the classic controller adapt to the conditions left by the tilting controller, and vice-versa.

The position profile is followed with a limited error, as shown in 23 and 25, and the switch can provide good performance.

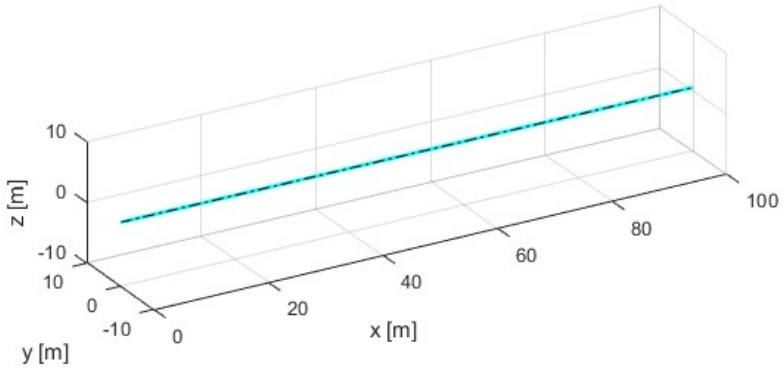
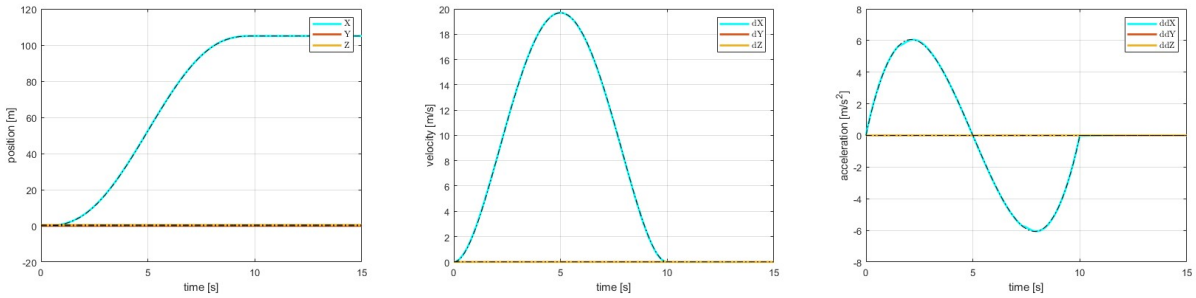


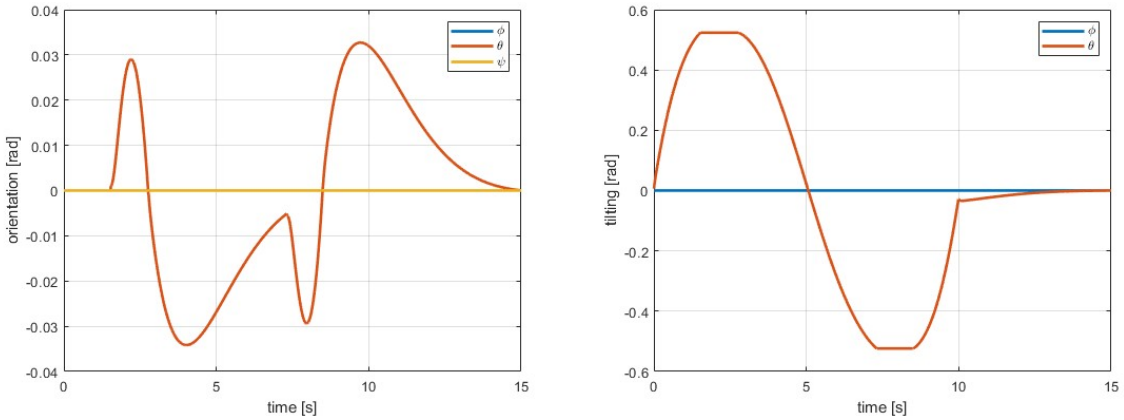
Figure 22: 3D path with hybrid controller



(a) (b) (c)

Figure 23: Cartesian position (a), velocities (b) and accelerations (c) with the hybrid controller

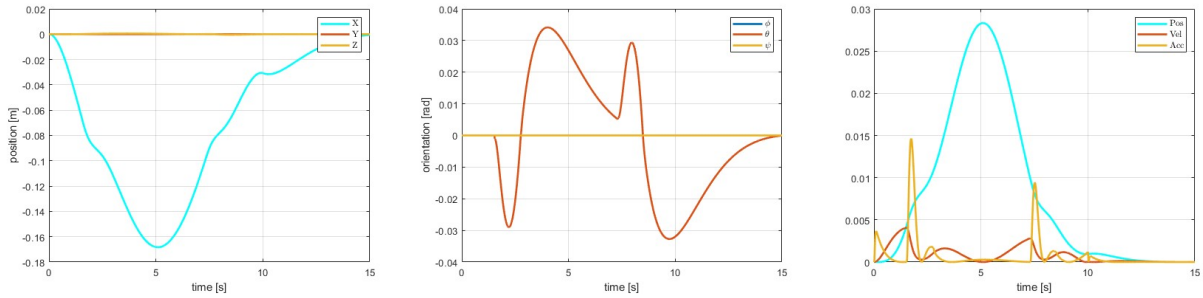
The body orientation tends to return to zero as soon as it is possible to return to the full actuation condition: through this mixed control, we maximize the time in which the body is not tilted, changing its orientation only when it is necessary about a small angle value, as seen in figures 24a and 24b.



(a) (b)

Figure 24: Body orientation (a) and tilting angles (b) with the hybrid controller

From these results, we can affirm to have effectively stabilized a trajectory that with the proposed controller was unfeasible, with the possibility to minimize the body orientation angles and partially achieve the goal of this new drone structure, maintaining at the same time stability.



(a) (b) (c)

Figure 25: Cartesian position (a), body orientation (b) and quadratic norm (c) errors with the hybrid controller

The execution of the hybrid controller with its switches does not introduce important disturbances in the vehicle's behavior. In 25a we can observe that in a trajectory that is 100 m long, the maximum positional error wrt the desired path is about 16 cm. The orientation error is kept under 0.04 rad despite the use of the body orientation to direct thrust (25b). Lastly, the quadratic error gives us a piece of significant information: thanks to the hybrid controller the magnitude of the velocity and acceleration error is kept below the values seen in the classic controller simulation.

5 Conclusions

The main objective of this report was to introduce a novel design for a quadcopter, which can overcome the limitations of underactuation that affect the performance of a conventional UAV. By adding two additional actuators that can adjust the orientation of the propellers along the x and y axes of the drone's body frame, we aimed to enhance the maneuverability and stability of the quadcopter.

We evaluated the effectiveness of our proposed design by applying a dynamic feedback linearization control technique, which transforms the nonlinear dynamics of the quadcopter into a linear system with simple integrators. We tested our controller on various trajectory tracking tasks, and demonstrated its superior capabilities for achieving precise lateral motion paths and regulating the body orientation.

However, we also acknowledged some drawbacks of this new structure, including the strict tilting joint limits that can compromise the above-mentioned advantages.

To address this issue, we proposed a hybrid control approach that combines our dynamic feedback linearization controller with a classical controller that does not rely on tilting mechanisms. We showed through simulations that this hybrid controller can effectively handle the transitions between different modes of operation and improve the performance of our quadcopter making it able to perform trajectories with desired body orientation useful to capture videos but also to perform fast trajectories to escape some dangerous situations.

References

[1] M. Odelga, P. Stegagno, and H. H. Bülthoff, "A fully actuated quadrotor uav with a propeller tilting mechanism: Modeling and control," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2016, pp. 306–311.

[2] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[3] S. A. Al-Hiddabi, "Quadrotor control using feedback linearization with dynamic extension," in *2009 6th International Symposium on Mechatronics and its Applications*, 2009, pp. 1–3.