Journal of Intelligent & Robotic Systems (2020) 100:531–574
https://doi.org/10.1007/s10846-020-01203-2

Trajectory Tracking for Aerial Robots: an Optimization-Based Planning and Control Approach

Jose Luis Sanchez-Lopez[1] · **Manuel Castillo-Lopez**[1] · **Miguel A. Olivares-Mendez**[1] · **Holger Voos**[1,2]

Received: 30 September 2019 / Accepted: 14 April 2020 / Published online: 22 July 2020

Abstract

In this work, we present an optimization-based trajectory tracking solution for multirotor aerial robots given a geometrically feasible path. A trajectory planner generates a minimum-time kinematically and dynamically feasible trajectory that includes not only standard restrictions such as continuity and limits on the trajectory, constraints in the waypoints, and maximum distance between the planned trajectory and the given path, but also restrictions in the actuators of the aerial robot based on its dynamic model, guaranteeing that the planned trajectory is achievable. Our novel compact multi-phase trajectory definition, as a set of two different kinds of polynomials, provides a higher semantic encoding of the trajectory, which allows calculating an optimal solution but following a predefined simple profile. A Model Predictive Controller ensures that the planned trajectory is tracked by the aerial robot with the smallest deviation. Its novel formulation takes as inputs all the magnitudes of the planned trajectory (i.e. position and heading, velocity, and acceleration) to generate the control commands, demonstrating through in-lab real flights an improvement of the tracking performance when compared with a controller that only uses the planned position and heading. To support our optimization-based solution, we discuss the most commonly used representations of orientations, as well as both the difference as well as the scalar error between two rotations, in both tridimensional and bidimensional spaces $SO(3)$ and $SO(2)$. We demonstrate that quaternions and error-quaternions have some advantages when compared to other formulations.

Keywords Trajectory tracking · Trajectory planning · Aerial robotics · Multirotor · UAV · MAV · Remotely operated vehicles · Mobile robots · Model predictive control · Optimization

1 Introduction

1.1 Motivation

Multiple studies, [57], foresee a great number of civilian applications of multirotor aerial robots (also called drones or Unmanned Aerial Vehicles, UAVs), such as their integration in smart cities [40], or aerial inspection [9], among others. Most of these applications are either under research and development as prototypes or they are still simply concepts. Only a few of them have already become a reality that some service provider companies are commercially exploiting by use the multirotor aerial vehicles in a remotely piloted way with a low-level of autonomy. Despite showing great potential, a high-level of autonomy is required to make most of the applications become a reality, as well as to increase the efficiency, usability, and safety of the already existing ones.

Recent advances in fully autonomous architectures and software frameworks like Aerostack [48, 49], have demonstrated the high level of autonomy [11] required to fully integrate aerial robots in daily use. Nevertheless, there are still multiple open research problems that limit the use of aerial robots. Two important ones are motion planning [19] and control [4, 35].

There are two different motion control tasks in robotics that sometimes are confused in the literature and need to be differentiated: path following and trajectory tracking. Paraphrasing [3], in path-following, the robot is required to converge to and follow a path that is specified without

✉ Jose Luis Sanchez-Lopez
joseluis.sanchezlopez@uni.lu

[1] Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust 29, University of Luxembourg, Avenue J. F. Kennedy, L-1855, Luxembourg, Luxembourg

[2] Faculté des Sciences, de la Technologie et de la Communication, University of Luxembourg, Luxembourg, Luxembourg

a temporal law. On the other hand, in trajectory-tracking, the robot has to reach and follow a time parameterized reference. A trajectory is, therefore, defined as a time parameterized reference (i.e., a geometric path with an associated timing law).

Similarly, it needs to be distinguished between two different motion planning tasks: path planning and trajectory planning. In path planning, the goal is to generate geometrically feasible (e.g. collision-free) paths, i.e. without time parametrization. In trajectory planning, not only geometrical feasibility is considered, but also kinematic and dynamic limits are taken into account to generate a trajectory.

To solve the problem of motion planning and control, we adopt a frequently seen approach in the literature,[26, 38, 46, 51], by combining three different components: (1) a collision-free geometric path planner, (2) a trajectory planner that computes a feasible trajectory from the previously given path, considering the restrictions of the kinematic and dynamic model of the robot, and (3) a controller for the trajectory tracking.

1.2 Problem Formulation and Objectives

In this work, we assume to have a geometrically feasible (e.g. collision-free) path given by a geometric path planner such as [51, 52]. This path, P, is defined as a discrete set of waypoints, W_i, that encode the desired position and heading of the aerial robot, as displayed in Fig. 1. The initial position of the robot is considered as the first waypoint.

A trajectory planner must generate a trajectory, L, used by a trajectory tracking controller as a reference, following an efficiency criterion. This trajectory consists of the relationship between time and the desired values of the position, heading of the aerial robot and all their derivatives, which the robot has to track. The trajectory has to pass through the waypoints and it must follow the given path, with a user-defined bounded maximum distance between the planned trajectory and the given path to avoid collisions with the obstacles of the environment, as a requirement for industrial-oriented applications. It has also to be feasible (kinematically and dynamically) by the robot.

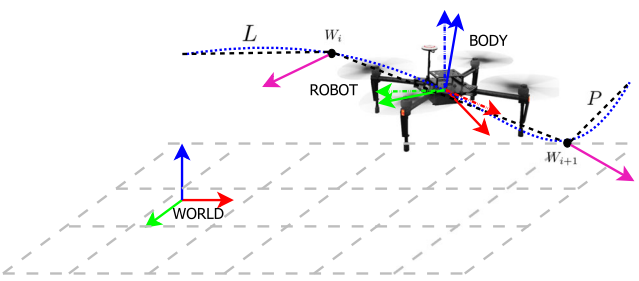


Fig. 1 Problem formulation. The desired path is represented with a black dashed line through the waypoints that encode the desired position (black circles) and desired heading (purple arrows). The planned trajectory is depicted by a blue dotted line

A trajectory tracking controller has the responsibility of the generation of the control commands that ensure the generated trajectory, L, is tracked by the aerial robot with the smallest deviation, correcting the effect of the disturbances through the estimated state of the aerial robot.

1.3 Contributions and Outline

In this paper, we present our latest advances in trajectory tracking. This work is the natural continuation of our previous study [50], where we presented our research on optimization-based trajectory planning. Back then, we validated our proposed trajectory planner with in-lab real flight experiments using a trajectory controller that combined a fuzzy-logic based feedback component with a feed-forward component that was taking advantage of the planned trajectory.

As the first contribution of this work, we present a multi-phase trajectory definition as a set of two different kinds of polynomials (i.e. acceleration/deceleration and constant velocity), more suitable for real applications such as inspection or package delivery, unlike other research-oriented aggressive maneuvers shown in the literature (e.g. [38]). This definition provides a higher semantic encoding of the trajectory which allows calculating, without loss of flexibility, an optimal solution but following a predefined simple profile. When comparing to our previous work, we have reformulated this definition, to reduce the number of parameters needed to describe the trajectory by implicitly including the continuity restrictions.

Secondly, we have reformulated our previous optimization-based trajectory planning solution from a given path. Our formulation is still based on total time minimization, instead of energy or snap. Our planner includes restrictions such as continuity of the trajectory (i.e. class C^m), limits on velocity, and higher-order derivatives, constraints in the waypoints, and maximum distance between the planned trajectory and the given path. Our new formulation considers the aforementioned restrictions as well as additional restrictions in the actuators of the aerial robot by using its dynamic model, relaxing the dependence of setting robot-feasible limits on the trajectory values (i.e. velocity, acceleration, jerk, etc.), and guaranteeing that the planned trajectory will be achievable by the robot.

Third, we present our optimization-based Model Predictive Controller (MPC) for trajectory tracking. Our new approach takes from the planner, at each sample time, all the magnitudes of the planned trajectory (i.e. position and heading, velocity, and acceleration) as well as the control command references computed using the planned trajectory, for all the steps in the prediction horizon. As confirmed by

our in-lab real flight experiments, our formulation has a better performance when compared with a controller that only uses the planned position and heading.

As the final contribution, to support our optimization-based solution in both the planner and controller, we analyze the most commonly used formulations to represent orientations and their kinematic relationships in the tridimensional space $SO(3)$ and in the bidimensional space $SO(2)$. We also include the mathematical formulations to compute difference between two rotations as well as the definition of a scalar error between two orientations, which is needed in optimization-based problems involving rotations. As discussed in the experimental part, our formulation uses quaternions and error-quaternions, as it has some advantages when compared to other formulations.

The remainder of the paper is organized as follows: Section 2 discuses the existing related works. Sections 3 and 4 includes the mathematical formulations to work with orientations in the tridimensional space $SO(3)$ and in the bidimensional space $SO(2)$. Section 5 describes the aerial robot model, whereas Section 6 defines the trajectory model used, to formulate later in Section 7 our proposed trajectory planner by means of an optimization problem. In Section 8 the model predictive trajectory tracking controller is presented. An evaluation and discussion of the results of this work are done in Section 9. Finally, Section 10 concludes the paper and points out some future lines of work. In addition to this, Appendices A and B provide extra information regarding the trajectory definition and the evaluation, respectively.

2 Related Work

The field of trajectory tracking from a given collision-free path is a recurrent topic found in the literature. Most of the existing works divide the trajectory tracking task into two parts: trajectory planning (Section 2.1), and trajectory tracking control (Section 2.2).

2.1 Trajectory Planning

Regarding the trajectory planning task, it is worth to mention two different kinds of works that unfortunately are often confused and mixed in the literature: (1) path smoothing and (2) trajectory planning. The first set of works, path smoothing, generate a smoother path given a desired (collision-free) path. The generated smooth path is computed following particular design criteria. In [58], a continuous curvature path smoothing algorithm for fixed-wing aerial robots is presented. In [26, 27], the authors propose a splines-based optimization to plan a path for multirotor aerial robots. In [27] the path is planned over a grid, whereas in [26], the path is planned with the help of Voronoi diagrams. The outcome of these works is not a trajectory, but simply a smooth path, i.e. there is no time dependence and therefore, velocity and higher-order derivatives are not computed. The main drawback of these approaches is that kinematic and dynamic constraints, e.g. velocity and acceleration bounds and continuity, are not guaranteed to be satisfied, and only the curvature of the path is considered.

If we consider only the trajectory planning problem itself, as the generation of time parameterized references given desired paths, different application domains should be analyzed to have a complete overview of the state of the art, since most of the ideas are exportable from one field to another. One of the first application domains and still a quite active one is comprised of the robotic manipulators: In [22], the authors generate smooth velocity and acceleration-bounded trajectories for robotic manipulators by replacing parts of the given path by closed-form time-optimal collision-free segments. In [16, 21, 36], the authors propose an analytical closed-form algorithm for the generation of velocity, acceleration, and jerk-bounded trajectories. The proposed trajectory is formed by a set of polynomials and is defined by different phases, i.e. acceleration increment, sustained acceleration, acceleration decrement, and constant velocity. The waypoints can include any arbitrary desired command, i.e. velocities and accelerations.

Trajectory planning for mobile robots is as well a quite active research field. In [12], the authors present a time-optimal algorithm for velocity planning of wheeled ground robots (e.g. autonomous driving cars). Their proposal discretize the trajectory and apply a closed-form optimization real-time capable method, considering kinematic constraints and some bounds on linear normal acceleration. In [37], the authors propose a kinematically feasible trajectory for fixed-wing aerial robots that connects a series of waypoints.

Multirotor aerial robots have some particular features that make them being different from another kind of mobile robots, as they can move in the 3D space, or they can reach any position in a holonomic way.

In [43], the authors propose a closed-form analytical trajectory planning approach to transition from one state to another state checking feasibility and minimizing the jerk. The planned trajectory is defined as a polynomial. [8] is an extension of the previous work for fully-actuated multirotor aerial robots. In [5–7], the authors propose a closed-form analytical trajectory planning approach from a given path. The trajectory has a multiple phases polynomial definition, like some other works mentioned above. In [5, 6], the proposed trajectory is jerk-bounded but jerk discontinuous. Despite being faster, and more suitable for real-time applications, having a closed-form analytical solution limits

the versatility of the trajectory planning, being impossible to add newer constraints such as the maximum distance between the trajectory and the original path, a dynamic model of the aerial robot, or certain requirements of the waypoints.

Another approach to trajectory planner for multirotor aerial robots is based on optimization algorithms. In [1, 38, 46], an optimization-based trajectory planning approach for multirotor aerial robot is presented. The trajectory is defined as a set of polynomials. Differential flatness is used to model the robot. Authors of [46] incorporate a collision-free check, while [1] simply impose some distance to path limitations in certain discrete trajectory parts. In [38, 46], the snap is minimized, whereas in [1] the energy (acceleration) is optimized but with the requirement to have a continuous snap. These works are very suitable for aggressive maneuvers, but the fact of not imposing distance to path restrictions difficult its usage in cluttered environments. Moreover, in [1, 38], the time to pass for the waypoints is imposed externally and not computed by the planner.

Interestingly, trajectory planning has been extended to multi-robot applications. In [23, 55], the authors combine a collision-free path planner with an optimization-based trajectory planner using a polynomial description of the trajectory.

Finally, it is worth to mention that there exist some approaches that combine in the same component the trajectory planner and the trajectory tracking controller, taking advantage of optimization-based control techniques.

For example, [42] propose a Model Predictive based Control (MPC), to make the multirotor aerial robot transition from the current state to another given state. Despite the proposed controller is generating a (discretized) trajectory for this transition, it cannot be fully compared with a trajectory planner since it does only consider one desired state and not a full path. The main drawback of these approaches is the tightness between the trajectory planner and the controller. This enforces the need for constantly re-planning at the control frequency which is unachievable for long and complex trajectories. Therefore, despite being theoretically feasible, it is not currently achievable in real-time, being this the main reason to have two different components, one for planning and another one for control.

2.2 Trajectory Tracking Control

Trajectory tracking control of multirotors has been an active topic of research during the last decade, as surveyed in [32]. Specifically, Model Predictive Control (MPC) approaches have shown to outperform traditional approaches in terms of tracking performance and robustness [41]. To increase the tracking performance, a cascade control hierarchy is often used, where feed-forward control commands are given to the MPC algorithm as control references. For instance, in [28] they design a hierarchical control structure for aerial robots where a linear quadratic regulator provides a steady-state reference to a model predictive controller to perform aggressive attitude tracking of aerial robots. Unlike our work, they don't incorporate heading control in the MPC approach, which limits its planning capabilities when the planned trajectory involves heading maneuvers. In [44] they present a unified trajectory optimization framework based on model predictive control, where a sequential linear quadratic trajectory optimization is performed to generate a steady-state reference state. However, they do not include control saturations nor path constraints. In [2, 53] they perform inversion-based position control and trajectory following for micro aerial vehicles, but they do not include control saturations nor path constraints since they employ a linear quadratic regulator for position control. In [29] linear and nonlinear MPC for trajectory tracking are compared, computing a feed-forward control reference from the planned acceleration. However, the heading is not controlled by the MPC algorithm, which leads to prediction errors when wide heading maneuvers are needed.

The use of unit quaternions for attitude representation is limited in the MPC literature on aerial robots. To properly employ this representation one must define integration and error functions based on the quaternion algebra. However, different works make use of standard algebra and a post-normalization as a workaround. In [18] the quaternion kinematics are considered in an MPC approach to include perception objectives into the optimal control problem. However, the attitude error is posed as a difference of quaternions, which is an ill-defined notion since unit quaternions are closed under composition, not sum or difference. Outside of the optimization-based control, there are different works with a proper quaternion-based error formulation. For instance, in [15] they formulate a quaternion-based error vector by extracting the eigenaxis rotation, and in [56] they induce error formulations from rotation matrices. However, these methods lack control saturations and path constraints.

3 Representation of Rotations in $SO(3)$

In this section we compile some mathematical properties relative to the representation of orientations in the tridimensional space $SO(3)$. Most of the information here presented has been carefully extracted and summarized from multiple sources that deeply address the previously mentioned problem. Some of the used sources include [20], [24], [31], [59], and specially [54].

The reader must note that this section has been included to create a self-contained paper that is easy to follow and understand, but it could be partially skipped if the reader is familiar with the topic. Special attention requires Section 3.3, which presents, as part of this work's contributions, two proposals on how to compute a scalar error between two orientation for optimization problems.

3.1 Orientation of a Rigid Body

The orientation of a rigid body, B, in an inertial reference frame, W, can be represented using multiple different formalisms.

One widely used formalism is the unit quaternion, also called rotation quaternion:

$$\boldsymbol{q} = \begin{bmatrix} q_w \\ \hdashline \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} q_w \\ \hdashline q_x \\ q_y \\ q_z \end{bmatrix} \tag{1}$$

where q_w is the scalar part, and $\boldsymbol{q}_v$ is the vectorial part of the quaternion.

Quaternions follow a well defined particular algebra which extends the two-dimensional complex numbers to four dimensions. Unit quaternions are a sub kind of quaternions that have a unitary norm, i.e. $\|\boldsymbol{q}\| = 1$ and that are used to represent orientations, being possible to use their algebra to easily carry out geometric transformations, e.g. rotations composition or rotation inversion. Therefore, unit quaternions represent rotations on the tridimensional space $SO(3)$ using four variables and one constraint. Unit quaternions allow having a continuous and singularity-free representation of the orientation of the body (unlike for instance Euler angles) with a reduced number of variables (unlike rotation matrices). Nevertheless, unit quaternions do not represent the rotation uniquely, being $\boldsymbol{q}$ and $-\boldsymbol{q}$, two valid representations of the same orientation.

The Euler's rotation theorem states that any rotation can be expressed as a single rotation of an angle ϕ about an axis $\boldsymbol{u}$. This rotation formalism is called axis-angle representation. The rotation axis $\boldsymbol{u} \in \mathbb{R}^3$ is a unitary vector. The rotation angle is defined as $\phi \in \mathbb{R}$, and therefore, there are infinite axis-angle values that correspond to the same rotation. We define the rotation angle $\psi \in \mathbb{R}_{[-\pi,\,\pi]}$, as the rotation angle that has the minimum absolute value of all the ones that encode the same orientation. Therefore, all the rotation angles with the value $\phi = \psi + 2\cdot\pi\cdot k$, for any integer k, $\forall k \in \mathbb{Z}$, encode exactly the same rotation. It is worth to highlight that the axis-angle representation, similarly than quaternions, requires four variables and one constraint to represent the tridimensional space $SO(3)$.

The rotation vector, also called Euler vector, is computed from the axis-angle representation as:

$$\boldsymbol{\phi} = \phi \cdot \boldsymbol{u} \tag{2}$$

Similarly than the axis-angle, infinite rotation vectors correspond to the same rotation. Concretely, all the rotation vectors of length $\phi = \psi + 2\cdot\pi\cdot k$, for any integer k, $\forall k \in \mathbb{Z}$, encode exactly the same rotation. The reader must note that rotation vectors only require three variables to represent rotations on the tridimensional space $SO(3)$, nevertheless, unlike quaternions, performing geometric operations with them is not straightforward.

The axis-angle representation can be calculated from the rotation vector as:

$$\phi = \|\boldsymbol{\phi}\| \tag{3}$$

$$\boldsymbol{u} = \frac{\boldsymbol{\phi}}{\|\boldsymbol{\phi}\|} \tag{4}$$

Nevertheless, as the reader may have noticed, it can only be computed in the case that $\phi \neq 0$, being otherwise undefined the rotation axis $\boldsymbol{u}$.

The unit quaternion can be calculated from the axis-angle rotation and the rotation vector, using the following relationships:

$$\boldsymbol{q} = \begin{bmatrix} q_w \\ \hdashline \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ \hdashline \sin\frac{\phi}{2}\cdot\boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \cos\frac{\|\boldsymbol{\phi}\|}{2} \\ \hdashline \sin\frac{\|\boldsymbol{\phi}\|}{2}\cdot\frac{\boldsymbol{\phi}}{\|\boldsymbol{\phi}\|} \end{bmatrix} \tag{5}$$

which is only valid for the rotation vector, in the case of $\phi = \|\boldsymbol{\phi}\| \neq 0$.

In the case of a small rotation, $\phi \approx 0$, then, the unit quaternion can be approximated to:

$$\boldsymbol{q} = \begin{bmatrix} q_w \\ \hdashline \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ \hdashline \sin\frac{\phi}{2}\cdot\boldsymbol{u} \end{bmatrix} \approx \begin{bmatrix} 1 \\ \hdashline \frac{1}{2}\cdot\phi\cdot\boldsymbol{u} \end{bmatrix} = \begin{bmatrix} 1 \\ \hdashline \frac{1}{2}\cdot\boldsymbol{\phi} \end{bmatrix} \tag{6}$$

It is important to highlight that it is required that $q_w = \cos\frac{\phi}{2} \geq 0$, i.e. $\phi \in [-\pi, \pi] + 2\cdot\pi\cdot k,\ k \in \mathbb{Z}$. Otherwise, we need to use the equivalent quaternion $-\boldsymbol{q}$.

The rotation vector and the axis-angle can be computed from Eq. 5 using the following expression:

$$\boldsymbol{\phi} = \underbrace{2\cdot\arctan\left(\frac{\|\boldsymbol{q}_v\|}{q_w}\right)}_{\phi}\cdot\underbrace{\frac{\boldsymbol{q}_v}{\|\boldsymbol{q}_v\|}}_{\boldsymbol{u}} \tag{7}$$

which is valid only if $\|\boldsymbol{q}_v\| \neq 0$.

In case that $\|\boldsymbol{q}_v\| \approx 0$, the small rotation approximation presented in Eq. 6 can be used, obtaining:

$$\boldsymbol{\phi} \approx 2\cdot\boldsymbol{q}_v \tag{8}$$

3.2 Difference Between Two Orientations

The rotation existing between two different orientations can be represented employing the quaternion difference. The

quaternion difference, $\delta\boldsymbol{q}$, encodes the difference between the two quaternions $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ and it is calculated as:

$$\delta\boldsymbol{q} = \boldsymbol{q}_a^* \otimes \boldsymbol{q}_b \tag{9}$$

where $\boldsymbol{q}_a^*$ is the conjugate of $\boldsymbol{q}_a$ and $\otimes$ is the quaternion product.

To overcome the fact that the quaternions use four variables to represent the tridimensional space $SO(3)$, the rotation vector of the quaternion difference, $\delta\boldsymbol{\phi}$, can be used following Eqs. 7 and 8.

The rotation vector obtained from a quaternion difference, applying the small rotation angle approximation presented in Eq. 6, is often called in the literature error-quaternion, $\delta\boldsymbol{\theta}$, as:

$$\delta\boldsymbol{q} = \begin{bmatrix} \delta q_w \\ \hdashline \delta\boldsymbol{q}_v \end{bmatrix} \approx \begin{bmatrix} 1 \\ \hdashline \frac{1}{2}\cdot\delta\boldsymbol{\theta} \end{bmatrix} \Leftrightarrow \delta\boldsymbol{\theta} \approx 2\cdot\delta\boldsymbol{q}_v \tag{10}$$

The approximation presented in Eq. 10, is often used for non small angles, $|\delta\phi| > 0$. In this case, it is needed to take into account the same consideration mentioned in Eq. 6 that $\delta q_w = \cos\frac{\delta\phi}{2} \geq 0$, and otherwise, we need to use the equivalent quaternion $-\delta\boldsymbol{q}$.

3.3 Scalar Error Between Two Orientations

When dealing with optimization problems, it is often needed to have a scalar magnitude, e.g. for the cost function, representing the difference between two orientations, e.g. the current and the desired orientations in control problems.

A first possibility is to compute the norm of the rotation vector difference, $\delta\boldsymbol{\phi}$, obtaining:

$$e_\phi = \|\delta\boldsymbol{\phi}\|^2 = \delta\boldsymbol{\phi}^T\cdot\delta\boldsymbol{\phi} = 4\cdot\arctan^2\left(\frac{\|\delta\boldsymbol{q}_v\|}{\delta q_w}\right) \tag{11}$$

The reader must note that this expression is valid for all the values of the quaternion $\delta\boldsymbol{q}$, no matter the value of $\|\delta\boldsymbol{q}_v\|$, unlike Eq. 7 and 8.

Another option is to use the error-quaternion, $\delta\boldsymbol{\theta}$, obtaining:

$$e_\theta = \|\delta\boldsymbol{\theta}\|^2 = \delta\boldsymbol{\theta}^T\cdot\delta\boldsymbol{\theta} = 4\cdot\delta\boldsymbol{q}_v^T\cdot\delta\boldsymbol{q}_v \tag{12}$$

The reader must note that in this case, no special precaution has to be adopted, unlike Eq. 10.

Other possibilities have been studied in the literature, as the function Ψ proposed in Eq. 8 of [34], which is proportional to our e_θ, or the function Ψ proposed in Eq. 9 of [33], which has a certain resemblance to our e_ϕ.

3.4 Time-Derivatives of the Orientation

The angular velocity of the rigid body in coordinates of the inertial reference frame, is defined as the time derivative of the rotation vector of the body:

$$\boldsymbol{\omega} = \frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}(\phi\cdot\boldsymbol{u}) \tag{13}$$

The relation between the unit quaternion that represents the orientation of a rigid body and its angular velocity is:

$$\dot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \frac{1}{2}\cdot\bar{\boldsymbol{\omega}}_B^W\otimes\boldsymbol{q} = \frac{1}{2}\cdot\boldsymbol{q}\otimes\bar{\boldsymbol{\omega}}_B^B \tag{14}$$

where $\boldsymbol{\omega}_B^W = \boldsymbol{\omega}$ represents the angular velocity of the body in coordinates of the inertial reference frame; $\boldsymbol{\omega}_B^B$ represents the angular velocity of the body in body coordinates; $\bar{\boldsymbol{\omega}}$ represents the pure quaternion associated to the angular velocity $\boldsymbol{\omega}$; and $\otimes$ is the quaternion product.

The l-th derivative of the orientation of the body in the inertial reference frame is given by:

$$\boldsymbol{\omega}^{(l-1)} = \frac{\mathrm{d}^{(l-1)}\boldsymbol{\omega}}{\mathrm{d}t^{(l-1)}} = \begin{bmatrix} \frac{\mathrm{d}\omega_x^{(l-1)}}{\mathrm{d}t^{(l-1)}} \\ \frac{\mathrm{d}\omega_y^{(l-1)}}{\mathrm{d}t^{(l-1)}} \\ \frac{\mathrm{d}\omega_z^{(l-1)}}{\mathrm{d}t^{(l-1)}} \end{bmatrix}, \quad \forall l\in\mathbb{N}, \quad l>1 \tag{15}$$

4 Representation of Rotations in *SO*(2)

The tridimensional space $SO(3)$ can be reduced to the bidimensional space $SO(2)$ in the case we only interested in considering the yaw angle, omitting the value of the other two angles. As presented in Section 5, this is the case, when using some differential flatness properties of multirotor aerial robots.

Along this section, we present some mathematical properties relative to the representation of orientations in $SO(2)$, by adapting the ones developed in Section 3. The mathematical tools introduced in this section represent the foundations of the following sections when dealing with orientations.

4.1 Orientation of a Rigid Body

In the case that we omit the value of the pitch and roll angles, retaining only the value of the yaw, the unit quaternion, $\boldsymbol{q}$, that represents the attitude of the body, B, in the inertial reference frame, W, presented in Eq. 1, has the following shape:

$$\boldsymbol{q} = \begin{bmatrix} q_w \\ \hdashline 0 \\ 0 \\ q_z \end{bmatrix} \tag{16}$$

The attitude of the rigid body can be, therefore, represented by the following simplified unit quaternion, $\tilde{\boldsymbol{q}}$:

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} q_w \\ \hdashline q_z \end{bmatrix} \tag{17}$$

and therefore, it requires two values and one restriction to represent rotations in $SO(2)$.

In the case of the angle-axis representation, the rotation axis is simplified to $\boldsymbol{u} = [0,\ 0,\ 1]^T$, and therefore the rotation vector is simplified to:

$$\boldsymbol{\phi} = \phi \cdot \boldsymbol{u} = \phi \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \phi \end{bmatrix} \tag{18}$$

and therefore, the rotation vector in $SO(2)$ is a scalar magnitude, $\tilde{\boldsymbol{\phi}} = \phi$.

As mentioned in Section 3, the rotation angle is defined as $\phi \in \mathbb{R}$, and therefore, there are infinite axis-angle values that correspond to the same rotation. Similarly, we extend the definition of the rotation angle $\psi \in \mathbb{R}_{[-\pi,\,\pi]}$, as the rotation angle that has the minimum absolute value of all the ones that encode the same orientation as all the infinite possible values of ϕ. Therefore, all the rotation angles with the value $\phi = \psi + 2 \cdot \pi \cdot k$, for any integer k, $\forall k \in \mathbb{Z}$, encode exactly the same rotation. This rotation angle, ϕ, is equivalent to the yaw angle used in aviation and aerial robotics.

We define the orientation value, $p_\psi \in \mathbb{R}$, as the rotation angle that encodes the total angular state of the rigid body, i.e. it encodes the total angle rotated by the body since its initial state has been defined. It is, therefore, a continuous real number, unlike the previously defined rotation angles ψ or ϕ.

Following Eq. 5, the simplified unit quaternion can be calculated from the rotation vector as:

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} q_w \\ \hdashline q_z \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{1}{2} \cdot \phi\right) \\ \hdashline \sin\left(\frac{1}{2} \cdot \phi\right) \end{bmatrix} \tag{19}$$

$$= \begin{bmatrix} \cos\left(\frac{1}{2} \cdot \psi\right) \\ \hdashline \sin\left(\frac{1}{2} \cdot \psi\right) \end{bmatrix} \tag{20}$$

The reader must note that q_w is always positive when computed using the angle ψ as $\psi \in \mathbb{R}_{(-\pi,\,\pi]}$, what is not always the case when computed using ϕ. Nevertheless, both simplified unit quaternion represent the same rotation.

The yaw angle, ψ, represented by a simplified quaternions, can be obtained from Eq. 20, as:

$$\psi = 2 \cdot \arctan\left(\frac{q_z}{q_w}\right) \tag{21}$$

Similarly, the simplified unit quaternion can be computed from the orientation value, p_ψ, as:

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} q_w \\ \hdashline q_z \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{1}{2} \cdot p_\psi\right) \\ \hdashline \sin\left(\frac{1}{2} \cdot p_\psi\right) \end{bmatrix} \tag{22}$$

The reader must note that the simplified quaternion $\tilde{\boldsymbol{q}}$ can be calculated from the orientation value p_ψ, but not the other way around. Equation 19 is, therefore, a surjection $\tilde{\boldsymbol{q}} = f(p_\psi)$.

Finally, combining Eq. 21 and 22, the yaw angle, ψ, can be computed from the orientation value, p_ψ, as:

$$\psi = 2 \cdot \arctan\left(\tan\left(\frac{1}{2} \cdot p_\psi\right)\right) \tag{23}$$

Similarly than Eq. 22, Eq. 23 is a surjection.

The reader must note that the simplified quaternion, $\tilde{\boldsymbol{q}}$, the yaw angle, ψ, and the orientation value, p_ψ, represent the same orientation (see Fig. 2). It is important to remember that the orientation value, p_ψ, cannot be calculated from the other two representations, while the other two representations can be calculated from any other representation.

4.2 Difference Between Two Orientations

Equation 9 for the quaternion difference is simplified to:

$$\delta\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{q}}_a^* \otimes \tilde{\boldsymbol{q}}_b = \begin{bmatrix} q_{aw} \cdot q_{bw} + q_{az} \cdot q_{bz} \\ \hdashline q_{aw} \cdot q_{bz} - q_{az} \cdot q_{bw} \end{bmatrix} = \begin{bmatrix} \delta q_w \\ \hdashline \delta q_z \end{bmatrix} \tag{24}$$

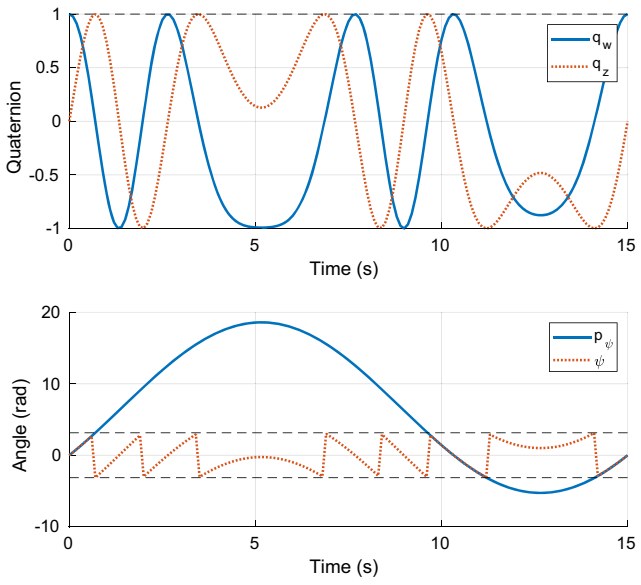


Fig. 2 The three different discussed representations in $SO(2)$ for the same rotations. Top: Simplified unit quaternion, being q_w the solid blue and q_z the dotted red. Bottom: p_ψ in solid blue, and ψ in dotted red

Combining Eq. 22 and 24, the quaternion difference is calculated in terms of the orientation value as:

$$\delta\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{q}}_a^* \otimes \tilde{\boldsymbol{q}}_b = \begin{bmatrix} \cos\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right) \\ \hdashline \sin\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right) \end{bmatrix} \tag{25}$$

Similarly, combining (20) and (24), the quaternion difference is calculated in terms of the yaw angle as:

$$\delta\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{q}}_a^* \otimes \tilde{\boldsymbol{q}}_b = \begin{bmatrix} \cos\left(\frac{1}{2}\cdot\left(\psi_b - \psi_a\right)\right) \\ \hdashline \sin\left(\frac{1}{2}\cdot\left(\psi_b - \psi_a\right)\right) \end{bmatrix} \tag{26}$$

The error-quaternion can be calculated for the simplified quaternions, following Eq. 10, as:

$$\delta\tilde{\boldsymbol{q}} = \begin{bmatrix} \delta q_w \\ \hdashline \delta q_z \end{bmatrix} \approx \begin{bmatrix} 1 \\ \hdashline \frac{1}{2}\cdot\delta\theta \end{bmatrix} \Leftrightarrow \delta\theta \approx 2\cdot\delta q_z \tag{27}$$

which, as mentioned before, requires $\delta q_w \geq 0$, otherwise, the quaternion $-\tilde{\boldsymbol{q}}$ must be used.

Combining (27) and (26), the error-quaternion is calculated in terms of the yaw angle, ψ, as:

$$\delta\theta \approx 2\cdot\sin\left(\frac{1}{2}\cdot\left(\psi_b - \psi_a\right)\right) \tag{28}$$

where, in this case, $\delta q_w = \cos\left(\frac{1}{2}\cdot\left(\psi_b - \psi_a\right)\right) \geq 0$ is always satisfied.

Similarly, combining (27) and (25), the error-quaternion is calculated in terms of the orientation value, p_ψ, as:

$$\delta\theta \approx \begin{cases} 2\cdot\sin\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right) \Leftrightarrow \delta q_w \geq 0 \\ -2\cdot\sin\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right) \Leftrightarrow \delta q_w < 0 \end{cases} \tag{29}$$

where $\delta q_w = \cos\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right)$.

The yaw angle difference $\delta\psi$ can be calculated as:

$$\delta\psi = 2\cdot\arctan\left(\frac{\delta q_z}{\delta q_w}\right) \tag{30}$$

$$= 2\cdot\arctan\left(\tan\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right)\right) \tag{31}$$

$$= 2\cdot\arctan\left(\tan\left(\frac{1}{2}\cdot\left(\psi_b - \psi_a\right)\right)\right) \tag{32}$$

Figure 3 plots the yaw angle difference, $\delta\psi$, and the error-quaternion, $\delta\theta$. As stated in Section 3.2, the error-quaternion approximation is often used for large rotation differences. The reader must note that this approximation is only equivalent to $\delta\psi$, in the environment where $\delta\psi \approx 0$.

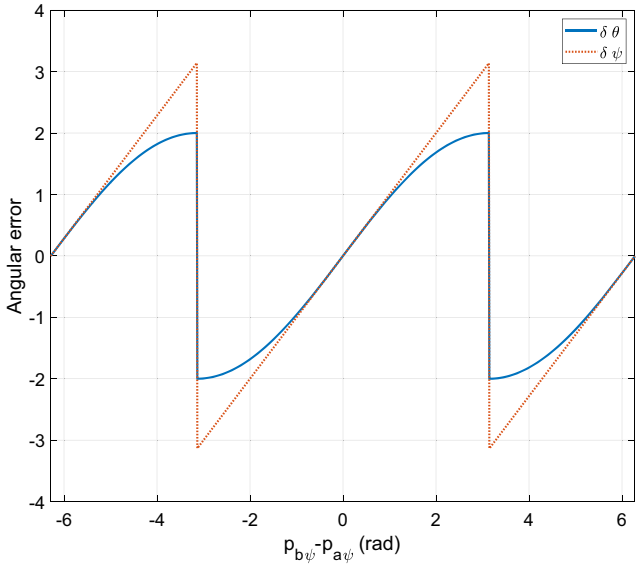


Fig. 3 Yaw angle difference, $\delta\psi$, in dashed red and error-quaternion, $\delta\theta$, in solid blue

4.3 Scalar Error Between Two Orientations

The norm of the rotation vector difference, $\delta\phi$, presented in Eq. 11, is simplified to:

$$e_\phi = \|\delta\phi\|^2 = 4\cdot\arctan^2\left(\frac{\|\delta q_z\|}{\delta q_w}\right) \tag{33}$$

$$= 4\cdot\arctan^2\left(\tan\left(\frac{1}{2}\cdot\left(\psi_b - \psi_a\right)\right)\right) \tag{34}$$

$$= 4\cdot\arctan^2\left(\tan\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right)\right) \tag{35}$$

The norm of the error-quaternion, $\delta\boldsymbol{\theta}$, presented in Eq. 12, is simplified to:

$$e_\theta = \|\delta\theta\|^2 = 4\cdot\delta q_z{}^T\cdot\delta q_z \tag{36}$$

$$= 4\cdot\sin^2\left(\frac{1}{2}\cdot\left(\psi_b - \psi_a\right)\right) \tag{37}$$

$$= 4\cdot\sin^2\left(\frac{1}{2}\cdot\left(p_{b_\psi} - p_{a_\psi}\right)\right) \tag{38}$$

Figure 4 plots both presented scalar rotation errors, the norm of the rotation vector difference, e_ψ, and the norm of the error-quaternion ,e_θ. Since the norm of the error-quaternion, it is only an approximation of the norm of the rotation vector difference, it does not have a good performance when the difference of the rotation angle is not $\delta\psi \approx 0$.

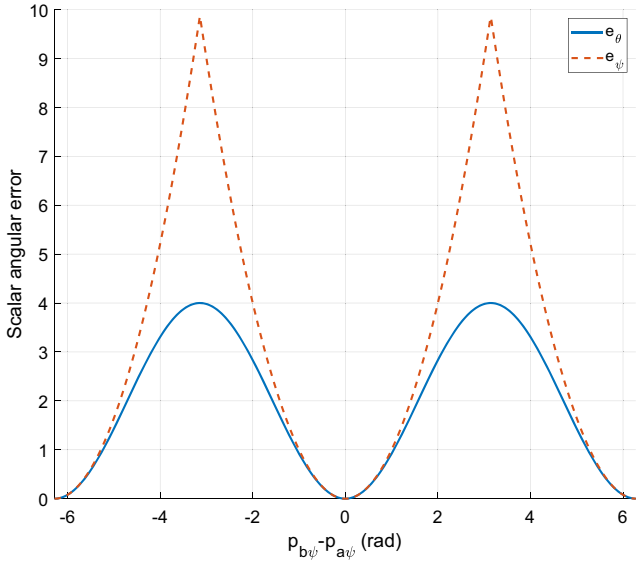


Fig. 4 e_ψ, in dashed red and e_θ, in solid blue

4.4 Time-Derivatives of the Orientation

The expression of the angular velocity of the rigid body in coordinates of the inertial reference frame, introduced in Eq. 13, is simplified to:

$$\boldsymbol{\omega} = \frac{\mathrm{d}\boldsymbol{\phi}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}(\phi \cdot \boldsymbol{u}) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\phi \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) = \frac{\mathrm{d}\phi}{\mathrm{d}t} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$
$$= \tilde{\omega} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \omega_\psi \end{bmatrix} \tag{39}$$

being the simplified angular velocity $\tilde{\boldsymbol{\omega}} = \tilde{\omega} = \omega_\psi$.

It is important to highlight that, from Eq. 39, the rotation angle, ϕ, is required to be continuous and derivable. To guarantee this derivability, we prefer to define the angular velocity in terms of the orientation value, p_ψ, as:

$$\omega_\psi = \frac{\mathrm{d}p_\psi}{\mathrm{d}t} \Rightarrow p_\psi = \int_{t_0}^{t_f} \omega_\psi \cdot \mathrm{d}t \tag{40}$$

Equation 14 that relates the unit quaternion with its angular velocity is simplified to:

$$\frac{\mathrm{d}\tilde{\boldsymbol{q}}}{\mathrm{d}t} = \frac{1}{2} \cdot \bar{\tilde{\boldsymbol{\omega}}}_B^W \otimes \tilde{\boldsymbol{q}} = \frac{1}{2} \cdot \tilde{\boldsymbol{q}} \otimes \bar{\tilde{\boldsymbol{\omega}}}_B^B \tag{41}$$

Equation 15, to calculate the l-th derivative of the orientation of the rigid body in the inertial reference frame is simplified to:

$$\tilde{\boldsymbol{\omega}}^{(l-1)} = \frac{\mathrm{d}^{(l-1)}\tilde{\boldsymbol{\omega}}}{\mathrm{d}t^{(l-1)}} = \left[\frac{\mathrm{d}\omega_\psi^{(l-1)}}{\mathrm{d}t^{(l-1)}}\right], \quad \forall l \in \mathbb{Z}, \quad l > 1 \tag{42}$$

Using the algebra of the quaternions, reduced to the case of the simplified quaternion, and considering its unity property when representing rotations, $\|\tilde{\boldsymbol{q}}\| = 1$, Eq. 41 is reduced to:

$$\frac{\mathrm{d}\tilde{\boldsymbol{q}}}{\mathrm{d}t} = \begin{bmatrix} \frac{\mathrm{d}q_w}{\mathrm{d}t} \\ \frac{\mathrm{d}q_z}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \cdot \omega_\psi \cdot \sqrt{1 - q_w^2} \\ \frac{1}{2} \cdot \omega_\psi \cdot \sqrt{1 - q_z^2} \end{bmatrix} \tag{43}$$

5 Aerial Robot Model

5.1 Reference Frames

The reference frames involved in the aerial robot modeling are represented in Fig. 1. The world reference frame, W, is arbitrary defined, attached to the ground and being its z-axis parallel to gravity. The body reference frame, B, is rigidly attached to the center of the robot, being its x-axis pointing to the front of the platform, and its z-axis perpendicular to the plane of the rotors. Finally, the robot horizontal reference frame, R, has its origin attached to the center of the platform and its x-axis is pointing to the front of the platform, but unlike the robot reference frame, its z-axis remains parallel to gravity, i.e. pitch and roll angles are zero. All the reference frames are right-handed.

5.2 Dynamical Model

The dynamics of multirotor aerial platforms are often modeled, [17, 38], considering their mechanical properties such as its mass, moment of inertia, distance from the axis of rotation of the rotors to the center of the quadrotor, drag coefficient of the propellers, etc., and the fact that it is possible to command directly the desired spinning speed of the rotors.

Since this model very accurately represents the dynamics of multirotor aerial platforms, its complexity is very high, being especially critical the accurate identification of the value of its mechanical properties. In addition, mo,[1] are equipped with a robust and efficient autopilot that provides embedded attitude and velocity controllers. A large amount of these industrial autopilots behave, from the final user point of view, like black-boxes with inputs-outputs and configuration parameters. It is therefore impossible to modify these autopilots further than what their manufacturers allow. It is highly uncommon that these autopilots offer the possibility to directly command, from a companion computer connected to it, the desired spinning speed of the rotors.

To overcome the aforementioned difficulties, exploiting the high-performance of the embedded autopilots, the dynamics of multirotor aerial platforms can be represented employing simplified models that include the dynamic

[1] DJI webpage: https://www.dji.com

response of the autopilot, as in [10, 53]. In these models, the control command given to the platform is the desired

$$\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \\ u_\psi \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_{xyz} \\ u_\psi \end{bmatrix} \tag{44}$$

where u_j represents the desired velocity of the aerial platform in the axis j of the reference frame attached to the aerial platform that is parallel to the ground (robot horizontal reference frame).

The dynamics of the aerial platform is represented with a simple first-order model as in [10]:

$$\boldsymbol{a}_R^R = \boldsymbol{K}_v \cdot \boldsymbol{v}_R^R + \boldsymbol{K}_{u_{xyz}} \cdot \boldsymbol{u}_{xyz} \tag{45}$$

$$\alpha_{\psi\,R}^{\;R} = \boldsymbol{K}_\omega \cdot \omega_{\psi\,R}^{\;R} + \boldsymbol{K}_{u_\psi} \cdot u_\psi \tag{46}$$

where $\boldsymbol{v}_R^R$, and $\boldsymbol{a}_R^R$ are the linear velocity and acceleration of the robot in robot coordinates. $\omega_{\psi\,R}^{\;R}$, and $\alpha_{\psi\,R}^{\;R}$ are the angular velocity and acceleration of the robot in robot coordinates. The robot model parameters are defined as:

$$\boldsymbol{K}_v = \begin{bmatrix} -\frac{1}{\tau_x} & 0 & 0 \\ 0 & -\frac{1}{\tau_y} & 0 \\ 0 & 0 & -\frac{1}{\tau_z} \end{bmatrix} \tag{47}$$

$$\boldsymbol{K}_{u_{xyz}} = \begin{bmatrix} \frac{k_x}{\tau_x} & 0 & 0 \\ 0 & \frac{k_y}{\tau_y} & 0 \\ 0 & 0 & \frac{k_z}{\tau_z} \end{bmatrix} \tag{48}$$

$$\boldsymbol{K}_\omega = \begin{bmatrix} -\frac{1}{\tau_\psi} \end{bmatrix} \tag{49}$$

$$\boldsymbol{K}_{u_\psi} = \begin{bmatrix} \frac{k_\psi}{\tau_\psi} \end{bmatrix} \tag{50}$$

where k_i and τ_i are the gain and time constants of each first-order velocity reference model. The remainder of the model of the aerial platform is calculated by using Kinematic relationships,[2] as follows:

$$\begin{cases} \dot{\boldsymbol{p}}_R^W = \boldsymbol{v}_R^W \\ \dot{\boldsymbol{v}}_R^W = \boldsymbol{a}_R^W \end{cases} \tag{51}$$

$$\begin{cases} \dot{\tilde{\boldsymbol{q}}}_R^W = \frac{1}{2} \cdot \omega_{\psi\,R}^{\;W} \otimes \tilde{\boldsymbol{q}}_R^W = \frac{1}{2} \cdot \tilde{\boldsymbol{q}}_R^W \otimes \omega_{\psi\,R}^{\;R} \\ \dot{\omega}_{\psi\,R}^{\;W} = \alpha_{\psi\,R}^{\;W} \end{cases} \tag{52}$$

$$\begin{cases} \boldsymbol{v}_R^W = \tilde{\boldsymbol{q}}_R^W \otimes \boldsymbol{v}_R^R \otimes \left(\tilde{\boldsymbol{q}}_R^W\right)^* = \boldsymbol{R}_R^W \cdot \boldsymbol{v}_R^R \\ \boldsymbol{a}_R^W = \tilde{\boldsymbol{q}}_R^W \otimes \boldsymbol{a}_R^R \otimes \left(\tilde{\boldsymbol{q}}_R^W\right)^* = \boldsymbol{R}_R^W \cdot \boldsymbol{a}_R^R \end{cases} \tag{53}$$

$$\begin{cases} \omega_{\psi\,R}^{\;W} = \omega_{\psi\,R}^{\;R} \\ \alpha_{\psi\,R}^{\;W} = \alpha_{\psi\,R}^{\;R} \end{cases} \tag{54}$$

[2] The notation v_A^B represents the value of the magnitude v of the system A represented in coordinates of the system B.

where $\boldsymbol{R}_R^W$ is the rotation matrix associated to the simplified quaternion $\tilde{\boldsymbol{q}}_R^W$. The reader should note the following relationships: $\boldsymbol{p}_R^W = \boldsymbol{p}_B^W$, $\boldsymbol{v}_R^W = \boldsymbol{v}_B^W$, and $\boldsymbol{a}_R^W = \boldsymbol{a}_B^W$.

The nominal state of the aerial robot platform can be defined using the different rotation conventions presented in Section 4, as one of the following:

$$\boldsymbol{x}_R = \left[\boldsymbol{p}_R^W,\ p_{\psi\,R}^{\;W},\ \boldsymbol{v}_R^R,\ \omega_{\psi\,R}^{\;R}\right]^T \tag{55}$$

$$= \left[\boldsymbol{p}_R^W,\ \psi_R^W,\ \boldsymbol{v}_R^R,\ \omega_{\psi\,R}^{\;R}\right]^T \tag{56}$$

$$= \left[\boldsymbol{p}_R^W,\ \tilde{\boldsymbol{q}}_R^W,\ \boldsymbol{v}_R^R,\ \omega_{\psi\,R}^{\;R}\right]^T \tag{57}$$

The reader should note that the state in Eqs. 55 and 56 has dimension 8, while the state in Eq. 57 has dimension 9, as the unit simplified quaternion, $\tilde{\boldsymbol{q}}_R^W$, comes along with one restriction.

The dynamical model of the aerial robot platform can be reformulated to the standard non-linear state space representation as $\dot{\boldsymbol{x}}_R = f(\boldsymbol{x}_R, \boldsymbol{u})$. This representation will be useful for the trajectory controller presented in Section 8.

We define the state difference, $\delta\boldsymbol{x}$, also called error-state, as the difference between two states, $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$, as:

$$\delta\boldsymbol{x} = \boldsymbol{x}_a \ominus \boldsymbol{x}_b \tag{58}$$

where $\ominus$ represents the difference operation between two states, what is a vectorial difference for all the elements of the state vector, except for the elements that represent rotations, in which case, we use the operations presented in Section 4.2.

The error-state has therefore, the following shape:

$$\delta\boldsymbol{x}_R = \left[\delta\boldsymbol{p}_R^W,\ \delta\tilde{\boldsymbol{q}}_R^W,\ \delta\boldsymbol{v}_R^R,\ \delta\omega_{\psi\,R}^{\;R}\right]^T \tag{59}$$

$$= \left[\delta\boldsymbol{p}_R^W,\ \delta\psi_R^W,\ \delta\boldsymbol{v}_R^R,\ \delta\omega_{\psi\,R}^{\;R}\right]^T \tag{60}$$

$$= \left[\delta\boldsymbol{p}_R^W,\ \delta\theta_R^W,\ \delta\boldsymbol{v}_R^R,\ \delta\omega_{\psi\,R}^{\;R}\right]^T \tag{61}$$

The reader should note that the error-state in Eqs. 60 and 61 has dimension 8, while the error-state in Eq. 59 has dimension 9, as the unit simplified quaternion, $\delta\tilde{\boldsymbol{q}}_R^W$, comes along with one restriction.

5.3 Differential Flatness

According to [17, 38], the dynamics of an underactuated multirotor aerial robot is differentially flat. This means that the states and the inputs appearing in its model can be written as algebraic functions of four carefully selected flat outputs and their derivatives.

A common choice of the flat outputs, [17, 38], are the position of the center of mass of the body of the aerial robot in world coordinates, $\boldsymbol{p}_B^W = \left[p_x,\ p_y,\ p_z\right]^T$, and its

continuous yaw (heading) angle, here represented as the orientation value, ${p_\psi}_R^W$, introduced in Section 4.1.

This property facilitates the generation of trajectories since any smooth trajectory (with reasonably bounded derivatives, due to actuator limits) in the space of flat outputs can be followed by the aerial robot.

5.4 Control Command References

Due to the differential flatness property presented in the previous Section 5.3, the inputs that appear in the model of an underactuated multirotor aerial robot, can be written as an algebraic function of the flat outputs and their derivatives. These inputs are the control command references needed to follow a given trajectory, and they can be computed by inverting the dynamical model of the aerial robot platform presented in Section 5.2 (45) and (46):

$$\boldsymbol{u}^r_{xyz} = \left(\boldsymbol{K}_{u_{xyz}}\right)^{-1} \cdot \left(\boldsymbol{a}^R_R - \boldsymbol{K}_v \cdot v^R_R\right) \tag{62}$$

$$u^r_\psi = \left(\boldsymbol{K}_{u_\psi}\right)^{-1} \cdot \left(\alpha_\psi{}^R_R - \boldsymbol{K}_\omega \cdot \omega_\psi{}^R_R\right) \tag{63}$$

Thus, we can penalize deviations from the optimal controls by defining the following control error:

$$\delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}^r \tag{64}$$

The reader must note that these control command references will never perform an accurate trajectory tracking when inputted to the real aerial platform. This is due to the fact that the real dynamics of the aerial robot platform is never going to be perfectly represented by the aforementioned dynamic model, on account of modeling simplifications and errors, as well as unconsidered external disturbances. Therefore, the trajectory tracking task requires a trajectory tracking controller compensating these errors, like the one presented in Section 8.

6 Trajectory Definition

6.1 General Description of the Trajectory

The trajectory, L, is defined as a piecewise set of n_s segments, $\boldsymbol{s}_i$, that depend on the time, t_i, which is delimited within a time range, $t_i \in [t_{i0},\ t_{if}]$:

$$L = \{\boldsymbol{s}_i(t_i), \quad t_i \in [t_{i0},\ t_{if}], \quad \forall i = \{1..n_s\}\}$$

The number of segments, n_s, of the trajectory, L, depends on the number of waypoints, n_w, of the given path, P.

Carrying out the following change on the time variable:

$$\tau_i = t_i - t_{i0}, \quad \tau_i \in [0,\ \Delta\tau_i = t_{if} - t_{i0}]$$

the trajectory, L, is redefined as:

$$L = \{\tilde{\boldsymbol{s}}_i(\tau_i), \quad \tau_i \in [0,\ \Delta\tau_i], \quad \forall i = \{1..n_s\}\}$$

All the segments, $\tilde{\boldsymbol{s}}_i(\tau_i)$, are defined in the space of the flat outputs, $\mathbb{R}^3 \times SO(2)$, defined in Section 5.3:

$$\tilde{\boldsymbol{s}}_i\,(\tau_i) : [0,\ \Delta\tau_i] \rightarrow \mathbb{R}^3 \times SO(2) \tag{65}$$

where

$$\tilde{\boldsymbol{s}}_i(\tau_i) = \begin{bmatrix} \boldsymbol{p}_{i,:}(\tau_i) \\ p_{i,\psi}(\tau_i) \end{bmatrix} = \begin{bmatrix} p_{i,x}(\tau_i) \\ p_{i,y}(\tau_i) \\ p_{i,z}(\tau_i) \\ p_{i,\psi}(\tau_i) \end{bmatrix}, \quad \forall i = \{1..n_s\} \tag{66}$$

The value of the orientation, $\tilde{\boldsymbol{q}}_{i,\psi}(\tau_i)$, is calculated following (22).

Each of the dimension, $p_{i,j}$ of the segments $\tilde{\boldsymbol{s}}_i$, being $j = \{x, y, z, \psi\}$, are defined as polynomial functions of order m_i, i.e. the $(m_i + 1)$-th derivative of the position, $p_{i,j}$, with respect to the time, τ_i, is zero:

$$p_{i,j}(\tau_i) = \sum_{k=0}^{k=m_i} \left(b_{i,j,k} \cdot \tau_i^k\right) \tag{67}$$

where $b_{i,j,k}$ are the coefficients of the polynomial.

The l_m-th time-derivative of the each polynomial is calculated by differentiation of Eq. 67 as:

$$\begin{aligned} p^{(l_m)}_{i,j}\,(\tau_i) &= \frac{\mathrm{d}^{l_m} p_{i,j}}{\mathrm{d}\tau_i^{l_m}} \\ &= \sum_{k=0}^{k=m_i} \left(b_{i,j,k} \cdot \left(\prod_{l=k-(l_m-1)}^{k} l\right) \cdot \tau_i^{k-l_m}\right) \end{aligned} \tag{68}$$

This definition of the trajectory allows to calculate a trajectory that is continuous up to n_d-th order, i.e. $T \in C^{n_d}$. In other words, all the segments and their derivatives up to the n_d-th order are continuous.

6.2 Continuity of the Trajectory

As mentioned in Section 6.1, the proposed definition of the trajectory allows to calculate a trajectory that is continuous up to n_d-th order, i.e. $T \in C^{n_d}$.

This continuity imposition, allows to make a distinction in the initial state of every segment: (1) the part that is given by the continuity restriction, $\boldsymbol{x}_{i,j,0:n_d}(0)$; and (2) the part that is not influenced by the continuity restriction, $\boldsymbol{x}_{i,j,(n_d+1):m_i}(0)$:

$$\boldsymbol{x}_{i,j,0:m_i}\,(0) = \begin{bmatrix} \boldsymbol{x}_{i,j,0:n_d}(0) \\ \boldsymbol{x}_{i,j,(n_d+1):m_i}(0) \end{bmatrix} \tag{69}$$

Similarly, the final state of the segment, $\boldsymbol{x}_{i,j,0:m_i}\,(\Delta\tau_i)$, has two parts, and it can be calculated, given its initial state (see Eq. 108 of Appendix A).

The initial state of the segment $i+1$, $\boldsymbol{x}_{i+1,j,0:n_d}(0)$ can be computed applying the continuity property to the final state of the segment i, $\boldsymbol{x}_{i,j,0:n_d}(\Delta\tau_i)$.

We have $\forall i = \{1..(n_s - 1)\}$, $\forall j = \{x, y, z, \psi\}$, and $\forall l = \{0..n_d\}$, the following expression:

$$p^{(l)}_{i+1,j}(0) = p^{(l)}_{i,j}(\Delta\tau_i) \tag{70}$$

and using the compact formulation (see Appendix A):

$$\boldsymbol{x}_{i+1,j,0:n_d}(0) = \boldsymbol{x}_{i,j,0:n_d}(\Delta\tau_i)$$

The reader must note that for the orientation, there is no need to use simplified quaternions when analyzing the continuity of the trajectory, as the orientation values $p_{i,\psi}(\Delta\tau_i)$ and $p_{i+1,\psi}(0)$ can be computed, and since they are continuous variables, they can be compared in the same way than the rest of the magnitudes.

6.3 Compact Description of the Trajectory

As mentioned in Section 6.1, every segment i, $\forall i = \{1..n_s\}$, of the trajectory T is represented by four polynomials, $\forall j = \{x, y, z, \psi\}$, of degree m_i. Every segment requires, therefore, the following variables to be completely defined:

- The time intervals of the segment, $\Delta\tau_i$.
- The coefficients of the polynomials, $\boldsymbol{b}_{i,j,:}$.

being the number of variables:

$$n_y = \sum_{\forall i} (4 \cdot (m_i + 1) + 1) \tag{71}$$

The coefficients of the polynomials can be calculated by using the initial state of the segment, $\boldsymbol{x}_{i,j,0:m_i}(0)$, (see Eq. 106 of Appendix A) and therefore, every segment can be completely defined by using:

- The time intervals of the segment, $\Delta\tau_i$.
- The initial state of the segment, $\boldsymbol{x}_{i,j,0:m_i}(0)$.

By imposing the continuity restriction presented in Section 6.2, every segment of the trajectory can be completely defined by using the following simplified set of variables:

- The time intervals of the segment, $\Delta\tau_i$.
- The initial state of the segment that is not influenced by the continuity restriction, $\boldsymbol{x}_{i,j,(n_d+1):m_i}(0)$.

being the number of variables in this case:

$$n_y = \sum_{\forall i} (4 \cdot \max((m_i - n_d), 0) + 1) \tag{72}$$

Thanks to this formulation that incorporates the continuity restrictions in the definition, the trajectory can be described more compactly with less number of variables.

The reader must note that the initial state of the segment that is influenced by the continuity restriction can be calculated with the previous segment as presented in Section 6.2. The only exception to this is the computation of the initial state of the first segment of the trajectory, which requires the state in the first waypoint.

6.4 Proposed Particular Description of the Trajectory

Our particular trajectory proposal has three segments between every two existing waypoints, which, as explained below, represent three different phases: acceleration, constant velocity, and deceleration. The number of segments is therefore calculated as:

$$n_s = 3 \cdot (n_w - 1) \tag{73}$$

We propose to have two kinds of polynomial segments, the ones that are connected to the waypoints, and the ones that are not. We have therefore two waypoint connected polynomials, per each intermediate one.

For the waypoint connected polynomials, we propose to force the seventh derivative, i.e. the lock, to be zero, obtaining for the position and the orientation-related value a 6th-degree polynomial, $m_i = 6$, with 7 coefficients per dimension and segment. These segments are used to represent the acceleration and deceleration movements of the robot.

For the intermediate polynomials, we force the second derivative, i.e. the acceleration, to be zero, obtaining for the position and the orientation-related value a 1st-degree polynomial, $m_i = 1$, with 2 coefficients per dimension and segment. These segments are used to represent a constant velocity movement of the robot.

As mentioned in Section 6.1, the definition of the trajectory allows to calculate a trajectory that is continuous up to n_d-th order, i.e. $T \in C^{n_d}$. Although our presented solution is general enough, we propose a continuous trajectory, up to third order, $n_d = 3$, that is, the third derivative, i.e. the jerk, is continuous but not derivable.

The advantage of this multi-phase trajectory definition as a set of two different kinds of polynomials (i.e. acceleration/deceleration and constant velocity), is that is more suitable for real applications such as inspection or package delivery, unlike other research-oriented aggressive maneuvers shown in the literature (e.g. [38]). This definition provides a higher semantic encoding of the trajectory which on the other hand, allows calculating, without loss of flexibility, an optimal solution but following a predefined simple profile.

7 Trajectory Planner

The trajectory planner calculates the suboptimal trajectory, L^*, defined by the parameters, $\boldsymbol{x}^*_f$, and $\Delta\boldsymbol{\tau}^*$, as illustrated

in Section 6, by solving the following single-objective nonlinear multivariable constrained minimization problem:

$$\boldsymbol{x}_f^*, \Delta\boldsymbol{\tau}^* = \underset{\boldsymbol{x}_f \in X, \Delta\boldsymbol{\tau} \in T}{\arg\min} \; (J(\Delta\boldsymbol{\tau}))$$

subject to:

Time feasibility:	$-\Delta\boldsymbol{\tau} \leq 0$
Continuity of the trajectory:	$c_s(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}) = 0$
Waypoints:	$w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = 0$
Dynamics of the traj.:	$\|v_T(\boldsymbol{x}_f, \Delta\boldsymbol{\tau})\|_2 - \|v_{max}\|_2 \leq 0$
Actuators of the robot:	$u_R(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}) - u_{max} \leq 0$
	$u_{min} - u_R(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}) \leq 0$
Distance to path:	$d(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) - d_{max} \leq 0$

The proposed optimization problem is detailed in the following Sections 7.1 to 7.5.

7.1 Optimization Variables

The optimization variables, $\boldsymbol{y}$, also called unknowns, gather all the parameters that describe the trajectory, L, introduced in Section 6.3, i.e.:

- The initial state of the polynomials of the segments that are set to free, $\boldsymbol{x}_f = \{x_{i,j}^{(k)}(0)\} \in X, \forall i = \{1..n_s\}, \forall j = \{x, y, z, \psi\}$, and $\forall k = \{(n_d + 1)..m_i\}$, where $m_i = 6$ for the case of waypoint connected polynomials, and $m_i = 1$ for the case of intermediate polynomials.
- The time intervals of the segments, $\Delta\boldsymbol{\tau} = \{\Delta\tau_i\} \in T$, $\forall i = \{1..n_s\}$.

being, therefore, the number of unknowns, $n_y = 27 \cdot (n_w - 1)$. All the unknowns are real numbers, i.e. $\boldsymbol{y} \in \mathbb{R}^{n_y}$.

7.2 Objective Function

The objective function, $J(\boldsymbol{y})$, also called cost function, that has to be minimized, is the total time of the trajectory tracking. It is calculated as

$$J(\Delta\boldsymbol{\tau}) = \sum_{\forall i} \Delta\tau_i \tag{74}$$

Other objectives like the minimization of the snap, as in [39], could be included in this objective function. Nevertheless, we will experimentally see in Section 9 that our proposed approach, also influences the snap while minimizing the total time.

7.3 Constraints

Two kinds of constraints, $c_{in}(\boldsymbol{y}) \leq 0$, and $c_{eq}(\boldsymbol{y}) = 0$, are included in the optimization problem formulation and are detailed below.

7.3.1 Time Feasibility

The time intervals of all the segments, $\Delta\tau_i$, must be feasible, i.e. cannot be negative:

$$\Delta\tau_i \geq 0, \quad \forall i = \{1..n_s\} \tag{75}$$

7.3.2 Continuity of the trajectory

As mentioned in Section 6, the trajectory has to be continuous up to n_d-th order, i.e. $T \in C^{n_d}$. In other words, all the segments and their derivatives up to the n_d-th order (in our case, $n_d = 3$) have to be continuous.

The continuity of the trajectory has been implicitly imposed in the trajectory definition (see Section 6.2). Including this constraint explicitly might be useful to overcome the possible issues appearing due to the fact that there are two kind of polynomial with different degree m_i.

We represent this set of restrictions as $c_s(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}) = 0$, and are computed following the equations presented in Section 6.2.

7.3.3 Waypoints

The trajectory has to pass through the waypoints of the given path, P. The reader must remember that the waypoint n, W_n, is characterized by is position, $\boldsymbol{p}_{W_n} = \left[p_{W_n,x}, \; p_{W_n,y}, \; p_{W_n,z}\right]^T$ and orientation, $\tilde{\boldsymbol{q}}_{W_n} = \left[q_{W_n,w}, \; q_{W_n,z}\right]^T$. We represent this set of restrictions as $w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = 0$.

For the position, we have $\forall j = \{x, y, z\}$, $\forall n \in P$, and $\forall i_-, i_+$ connecting polynomials, the following expressions:

$$p_{i_-,j}\left(\Delta\tau_{i_-}\right) = p_{W_n,j}$$
$$p_{i_+,j}\left(0\right) = p_{W_n,j}$$

Obtaining:

$$w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = p_{i_-,j}\left(\Delta\tau_{i_-}\right) - p_{w_n,j} = 0 \tag{76}$$
$$w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = p_{i_+,j}\left(0\right) - p_{w_n,j} = 0 \tag{77}$$

The value of $p_{i_-,j}\left(\Delta\tau_{i_-}\right)$ and $p_{i_+,j}\left(0\right)$ can be calculated by combining (89) and (106) of Appendix A.

Similarly, for the orientation, we have:

$$\tilde{\boldsymbol{q}}_{i_-,\psi}\left(\Delta\tau_{i_-}\right) = \tilde{\boldsymbol{q}}_{w_n}$$
$$\tilde{\boldsymbol{q}}_{i_+,\psi}\left(0\right) = \tilde{\boldsymbol{q}}_{w_n}$$

As presented in Section 4.2, we have two options to compute the reduced dimensionality difference between two rotation:

On the one hand, we can use the yaw angle difference, $\delta\psi$, defined in (30):

$$w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = f_{\delta\psi}\left(\tilde{\boldsymbol{q}}_{i_-,\psi}\left(\Delta\tau_{i_-}\right), \tilde{\boldsymbol{q}}_{w_n}\right) = 0 \quad (78)$$

$$w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = f_{\delta\psi}\left(\tilde{\boldsymbol{q}}_{i_+,\psi}\left(0\right), \tilde{\boldsymbol{q}}_{w_n}\right) = 0 \quad (79)$$

On the other hand, we can use the error-quaternion, $\delta\theta$, defined in Eq. 27:

$$w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = f_{\delta\theta}\left(\tilde{\boldsymbol{q}}_{i_-,\psi}\left(\Delta\tau_{i_-}\right), \tilde{\boldsymbol{q}}_{w_n}\right) = 0 \quad (80)$$

$$w(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) = f_{\delta\theta}\left(\tilde{\boldsymbol{q}}_{i_+,\psi}\left(0\right), \tilde{\boldsymbol{q}}_{w_n}\right) = 0 \quad (81)$$

Similarly than before, the value of $\tilde{\boldsymbol{q}}_{i_-,\psi}\left(\Delta\tau_{i_-}\right)$ and $\tilde{\boldsymbol{q}}_{i_+,\psi}\left(0\right)$ can be calculated by combining (89) and (106) of Appendix A.

The reader must note that one of the (76) and (77), and one of the (78) and (79), or (80) and (81), are redundant and therefore might be omitted for simplicity.

7.3.4 Dynamics of the trajectory

Certain limits need to be imposed on the dynamics of the trajectory, i.e. limit on velocity, acceleration, jerk, snap, crackle, pop, ... This may be needed due to requirements of the application (e.g. tracking a trajectory with a certain maximum velocity), or due to an incomplete (or too simple) model of the dynamics of the robot that does not consider higher-order derivatives or non-linear limits, as mentioned in Section 5.3.

These limits are represented as

$$\|v_T(\boldsymbol{x}_f, \Delta\boldsymbol{\tau})\|_2 \leq \|v_{max}\|_2$$

Which is equivalent to:

$$c_T(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, v_{max}) = \|v_T(\boldsymbol{x}_f, \Delta\boldsymbol{\tau})\|_2^2 - \|v_{max}\|_2^2 \leq 0 \quad (82)$$

where $\|v_T(\boldsymbol{x}_f, \Delta\boldsymbol{\tau})\|_2^2$ can be calculated by combining (97) and (122), with Eq. 106 of Appendix A.

7.3.5 Actuators of the robot

The actuators of the robots have physical limits that cannot be exceeded.

The control commands, $u_R(\boldsymbol{x}_f, \Delta\boldsymbol{\tau})$, are calculated following the dynamic model of the robot expressed in Eqs. 62 and 63, and the following constraints are applied:

$$u_{min} \leq u_R(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}) \leq u_{max} \quad (83)$$

Which is equivalent to

$$u_R(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}) - u_{max} \leq 0 \quad (84)$$

$$u_{min} - u_R(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}) \leq 0 \quad (85)$$

7.3.6 Linear distance to path

The euclidean linear distance between the given path and the position variables of the trajectory has to be lower than the value, d_{max}.

We represent this restriction as $d(\boldsymbol{x}_f, \Delta\boldsymbol{\tau}, P) - d_{max} \leq 0$, and it is calculated as:

$$\left|\left|\boldsymbol{p}_{i,:}\left(\tau_i\right) - \boldsymbol{p}_{P_{n,n+1}}\right|\right|_2 \leq d_{max} \quad (86)$$

$\forall\tau_i \in [0, \Delta\tau_i]$, $\forall i = \{1..n_s\}$, and for all the position subpaths, $\boldsymbol{p}_{P_{n,n+1}}$, that form the complete path P. The value $\boldsymbol{p}_{i,:}\left(\tau_i\right)$ can be calculated by combining (111) and (106) of Appendix A.

The reader must note that this restriction might only be applied to any user-defined sub-path of the given path P.

7.4 Initialization

The initialization of the unknowns is essential for the fast convergence of the optimization algorithm to a local minimum. The initial value of the unknowns, $\boldsymbol{y}_0$, has to be feasible from the optimization problem point of view, i.e. it has to fulfill the constraints described in Section 7.3.

We propose an initial trajectory, L_0, that exactly follows the given path, P, by setting to zero the velocity and all the continuous higher-order derivatives in the waypoints. This means that, between every two waypoints, the three following stages take place: acceleration from zero velocity, constant velocity movement, and deceleration to zero velocity. By definition, this initial trajectory pass through all the waypoints, and the linear distance to the path is zero. The acceleration and deceleration stages are carried out following the limits on the dynamics and actuators of the robot.

The aforementioned initial trajectory, L_0, is, therefore, feasible from the optimization problem point of view. It can be calculated analytically but for the sake of brevity, a complete expound of its computation method is omitted.

7.5 Recursive sequential optimization

To boost the computation of suboptimal trajectory, the aforementioned large optimization problem can be divided into a set of small optimization problems that can be solved sequentially.

The full path is sequentially divided into sub-paths of three waypoints, solving the optimization problem for only these three waypoints. Once the trajectory is computed for these three waypoints, we advance to the next waypoint of the path, and we solve the optimization problem for the sub-path formed by the previous last two waypoints and the new waypoint. The computed trajectory in every iteration

of the sequence updates the value of the initial trajectory calculated as mentioned in Section 7.4, which is used in the following sequence as the initial sub-trajectory of the small optimization problem. This process continues until the last waypoint is reached.

In order to improve the optimality of the computed trajectory, the full sequential optimization problem is solved again in a recursive way, always using the last computed trajectory, as the initial trajectory of the optimization problem.

A later joint optimization of the full trajectory could be done, using the last computed trajectory, as the initial trajectory of the optimization problem, to improve even more the optimality of the computed trajectory.

8 Trajectory Tracking Controller

To maximize the trajectory tracking performance, we design a Model Predictive Control (MPC) approach that, each sample time, obtains the control commands $\boldsymbol{u}$ by solving the following optimal control problem (OCP):

$$\min_{\boldsymbol{u}(t),\boldsymbol{x}(t)} J = \int_0^T \left(\|\delta \boldsymbol{x}_R(t)\|_P^2 + \|\delta \boldsymbol{u}(t)\|_Q^2 + ks^2 \right) dt \quad (87a)$$

$$s.t. \quad \dot{\boldsymbol{x}}_R(t) = f(\boldsymbol{x}_R(t), \boldsymbol{u}(t)) \quad (87b)$$

$$\|\boldsymbol{q}\|^2 = 1 - s \quad (87c)$$

$$\boldsymbol{u}_{min}(t) \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max}(t) \quad (87d)$$

where $\boldsymbol{x}_R$ and $\delta \boldsymbol{x}_R$ are the state and the error-state of the system as defined in Eqs. 57 and 61 respectively. Similarly, the control commands $\boldsymbol{u}$ and $\delta \boldsymbol{u}$ are defined in Eqs. 57 and 64 respectively. The state-space formulation of the dynamical model of the aerial robot is included in Eq. 87b, as developed in Section 5.2. The weighting matrices P and Q, and the weighting scalar k are tuning parameters for the desired objective function (87a). As discussed by [13], Eq. 87c needs to be included to prevent large violation of the quaternion's norm during optimization. In this work, we introduce it as a soft constraint [30], where the slack variable s minimizes the deviations of the quaternion from being unitary.

As a result, the MPC produces a predicted commanded trajectory $\boldsymbol{x}$ and control commands $\boldsymbol{u}$ that minimizes the tracking performance along the horizon T. The feasibility of this commanded trajectory is ensured through the dynamical model constraint (87b) and the control saturations (87d).

One of the key facts of this formulation is to include $\delta \boldsymbol{u}(t)$ instead of $\boldsymbol{u}(t)$ in the cost function to minimize the deviations from the optimal control command references (computed given the planned trajectory following Eqs. 62 and 63), which improves the tracking performance for two reasons. First, the controller follows feasible time-optimal control references instead of stabilizing ones. Second, the feasible evolution of the control reference improves the convergence of the optimization problem, since the state of the system is closer to the optimal one.

The other key fact is the incorporation of the quaternion algebra developed in Section 4, concretely the results of Eq. 36. This provides a singularity-free attitude control, whose rotation-matrix equivalent has been shown to be asymptotically stable for large attitude deviations [34].

To solve the OCP we employ direct optimization, where the problem is discretized through a multiple-shooting algorithm over N steps with a 4th order Runge-Kutta integration of the dynamics (87b). This integration scheme The resulting nonlinear program (NLP) is solved through a sequential quadratic programming (SQP) algorithm as described in [14].

9 Evaluation and Results

9.1 Evaluation methodology

The validation of the proposed trajectory tracking approach is done considering two different aspects.

On the one hand, in Section 9.3, we illustrate and evaluate the proposed trajectory planner. We analyze both qualitatively and quantitatively the optimization process of the trajectory planner together with the properties of the calculated trajectories. Additionally, we compare the two different presented options to compute the reduced dimensionality difference between two rotation when applying the waypoints restriction of the trajectory planner (discussed in Section 7.3.3).

We use the concept of the energy value, ν_{energy}, applied to different magnitudes, ν, of the trajectory:

$$\nu_{energy} = \int_0^{t_f} \|\nu(t)\|^2 \cdot \mathrm{d}t \quad (88)$$

On the other hand, we exemplify and evaluate the proposed trajectory tracking controller in Section 9.4 by means of in-lab real flight experiments. We study both qualitatively and quantitatively its different performance when taking advantage of the whole output of the trajectory planner (i.e. pose and higher-order derivatives, and control command references) and when only considering the desired pose.

We use the following well-known four error metrics:

- Mean Squared Error (MSE)
- Root Mean Square Error (RMSE)
- Mean Absolute Error (MAE)
- Max Absolute Error (MaAE)

Table 1 List of waypoints of the first path used for the evaluation. The path can be visualized in Fig. 5

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8
p_x (m)	−1.35	1.35	1.35	−1.35	1.35	1.35	−1.35	−1.35
p_y (m)	−1.35	−1.35	1.35	1.35	−1.35	1.35	1.35	−1.35
p_z (m)	1.25	1.25	1.25	1.25	2	2	2	1.25
ψ (°)	0	−90	180	90	−90	180	90	0

We have defined two paths for evaluation purposes. Both paths are limited by the dimensions of our flying arena that are 5.5 × 5.0 × 4.5 m (W × L × H).

The first path is defined by the 8 waypoints presented in Table 1, and can be visualized in Fig. 5 This path emulates a kind of spiral in position, with a constant rotation on the desired robot heading along with the waypoints. This spiral, apart from evolving in the horizontal coordinates, has also changes both up and down in the vertical coordinate z.

The second path used for the evaluation is defined by the 10 waypoints listed in Table 2 and it is shown in Fig. 6. This path aims to have a relatively large number of waypoints with a high complexity in the available reduced space, combining in some cases changes between waypoints in just one or two dimensions in a relatively short distance (e.g. W_1 to W_2 or W_2 to W_3), together with changes up to the four dimensions in larger distances (e.g. W_5 to W_6 or W_6 to W_7).

In both cases, we assume the aerial robot to be initially hovering with a pose that coincides with the first waypoint.

For comparison purposes, our trajectory planner has been configured with two different maximum linear distance to the path, d_{max}, (see Section 7.3.6), and with four different sets of limits on the dynamics of the trajectory, $\|v_{max}\|_2$, (see Section 7.3.4). The two maximum linear distance to the path, indicated in Table 3, are named accurate and inaccurate. The four sets of limits on the dynamics of the trajectory, summarized in Table 4, are called Slow, Medium-Slow, Medium-Fast, and Fast.

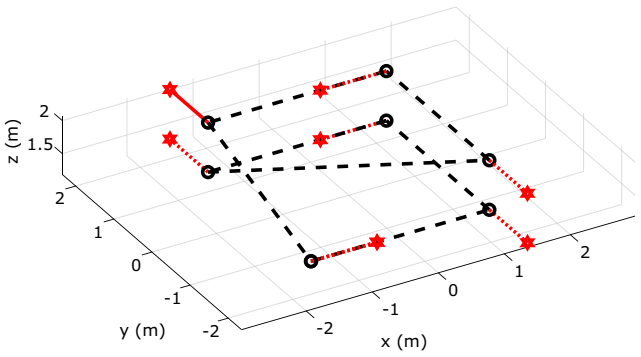


Fig. 5 3D view of the first path used for the evaluation. The path is represented with a dashed black line, being its waypoints, listed in Table 1, represented by a circle (position) and a red arrow (heading)

The total number of trajectories considering the two paths and all the previously mentioned configuration parameters combinations is 16. This number is doubled to 32 when taking into account that we have two different options to compute the reduced dimensionality difference between two rotation when applying the waypoints restriction of the trajectory planner, as discussed in Section 7.3.3.

Despite the limits on the dynamics of the trajectory may seem relatively small, the reader must note that they are quite challenging when considering the reduced size of our flying arena and the comparatively large size of our aerial robot, presented in Section 9.2.

9.2 Experimental setup

Our aerial robot platform is a DJI Matrice 100[3] quadrotor (see Fig. 7). It is equipped with a DJI N1 flight controller, that does not only stabilizes the platform but also provides a velocity controller that uses only onboard sensors as feedback (including a DJI Guidance[4]). This autopilot allows us to input a command in terms of the desired velocity of the platform (i.e. linear velocity and heading velocity, both in robot coordinates), unlike [17, 38], where the control commands are the spinning velocity of the motors of the aerial platform. Moreover, our aerial robot platform is equipped with some extra sensors and a companion computer.

The reader might note that our aerial robot (see Fig. 7) is considerably larger and heavier (Size: 890 × 890 × 340 mm^3, MTOW: 3600 g.) than the aerial platforms used in works like [1, 38, 46] that focus on aggressive maneuvering (e.g. AscTec Hummingbird.[5] Size: 540 × 540 × 85.5 mm^3, MTOW: 710 g.). Moreover, the reader must take into consideration, as previously presented in Section 9.1, the substantially small size of our flying arena when compared with the size of our aerial robot (our flying arena is only about 5 times larger than the aerial robot). Figure 8 shows our aerial robot flying autonomously inside our flight arena during an experimental validation test.

[3] https://www.dji.com/matrice100

[4] https://www.dji.com/guidance

[5] Source: http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird/

Table 2 List of waypoints of the second path used for the evaluation. The path can be visualized in Fig. 6

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}
p_x (m)	−1.5	0	1.5	1.5	0	−1.5	1.5	−1.5	−1.5	−1.5
p_y (m)	−1.5	−1.5	0	1.5	1.5	1.5	−1.5	1.5	−1.5	−1.5
p_z (m)	1.25	1.25	1.25	1.25	1.25	1.25	2	2	2	1.25
ψ (°)	0	45	45	90	135	180	0	−90	0	0

The values of the parameters k_j and τ_j presented in Section 5.2, have been calculated empirically, following the procedure described in [10], and are shown in Table 5.

Both the trajectory planner and the trajectory tracking controller have been configured with the control command (i.e. actuator) limits (see Sections 7.3.5 and 8) presented in Table 6. The reader must note that we have set the trajectory controller limits higher than the planner ones, to let the controller a margin to compensate model errors and disturbances and therefore improve the performance of the trajectory tracking.

Figure 9 shows the proposed system architecture for the experimental evaluation. All the components, except the trajectory planner, have been implemented in C++ using ROS [45] as middleware.

Our trajectory planner is executed offline and we have implemented it in MATLAB, using the single-objective nonlinear multivariable constrained minimization solver provided by the function *fmincon*. It has been configured to use the interior-point (IP) method, computing the Hessian by a dense quasi-Newton approximation. The trajectory tracking controller has been implemented using ACADO Toolkit [25], which exports efficient C code to solve the problem (87) efficiently through embedded integrators and sequential quadratic programming (SQP).

We collect information about the state of our aerial platform through an Optitrack motion capture system installed in our flying arena, which provides the measurements of its position and orientation at a frequency of 200 Hz. The state of the robot (i.e. pose and velocity) is estimated using [47].

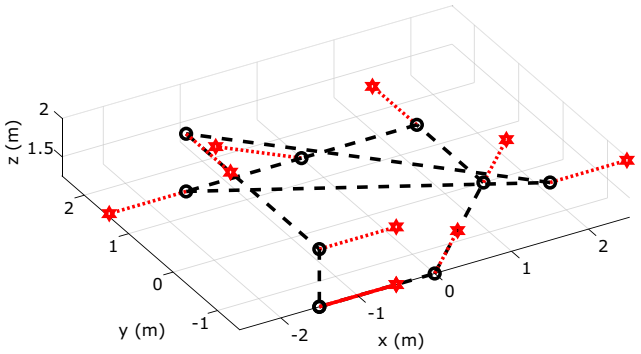


Fig. 6 3D view of the second path used for the evaluation. The path is represented with a dashed black line, being its waypoints, listed in Table 2, represented by a circle (position) and a red arrow (heading)

9.3 Trajectory planning results

In this section, we illustrate and evaluate the proposed trajectory planner, analyzing the optimization process and the properties of the calculated trajectories. Additionally, we compare the two different presented options to compute the reduced dimensionality difference between two rotation when applying the waypoints restriction of the trajectory planner (discussed in Section 7.3.3).

For the first part of this section, the 32 possible trajectories presented in Section 9.1 have been computed using the proposed trajectory planner until the objective has converged to a stable value, or until a maximum number of iterations (i.e. 100) has been reached.

As described in Section 7, the trajectory planner consists of an optimization process that computes a feasible trajectory that minimizes the total time of trajectory. The initial trajectory is computed analytically, and modified over the iterations carried out by the optimizer until the optimum trajectory is found. Figure 10 illustrates this optimization process displaying the first 4 waypoints of the first path, configured with an inaccurate maximum linear distance to the path, and using the error-quaternion, $\delta\theta$, to compute the difference between two rotation.

It can be seen in Fig. 10 that the initial trajectory (depicted in solid red) is coincident with the path (in solid black). The intermediate trajectories (in dashed cyan) computed at different iterations of the optimization process are calculated by increasing the maximum values of the trajectories (e.g. distance to the path) to minimize the total trajectory time. The final optimum computed trajectory (in solid blue) is the one that minimizes the total trajectory time.

Table 3 The two configuration parameters of the trajectory planner for the maximum linear distance to the path, d_{max}

	accurate (m)	inaccurate (m)
d_{path}	0.05	0.5

Table 4 The four set of configuration parameters of the trajectory planner for the limits on the dynamics of the trajectory, $\|v_{max}\|_2$

SLOW (S)					
$v^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-1}})$	$a^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-2}})$	$j^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-3}})$	$s^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-4}})$	$c^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-5}})$	$p^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-6}})$
1	2	6	15	90	600
$\omega_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-1}})$	$\alpha_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-2}})$	$j_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-3}})$	$s_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-4}})$	$c_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-5}})$	$p_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-6}})$
1	2	6	15	90	600
MEDIUM-SLOW (MS)					
$v^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-1}})$	$a^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-2}})$	$j^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-3}})$	$s^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-4}})$	$c^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-5}})$	$p^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-6}})$
1.5	3	9	27	135	810
$\omega_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-1}})$	$\alpha_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-2}})$	$j_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-3}})$	$s_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-4}})$	$c_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-5}})$	$p_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-6}})$
1.5	3	9	27	135	810
MEDIUM-FAST (MF)					
$v^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-1}})$	$a^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-2}})$	$j^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-3}})$	$s^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-4}})$	$c^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-5}})$	$p^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-6}})$
1.75	3.5	11	35	145	880
$\omega_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-1}})$	$\alpha_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-2}})$	$j_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-3}})$	$s_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-4}})$	$c_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-5}})$	$p_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-6}})$
1.75	3.5	11	35	145	880
FAST (F)					
$v^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-1}})$	$a^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-2}})$	$j^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-3}})$	$s^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-4}})$	$c^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-5}})$	$p^W_{R\|W}$ $(\mathrm{m{\cdot}s^{-6}})$
2	4	12	40	155	900
$\omega_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-1}})$	$\alpha_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-2}})$	$\iota_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-3}})$	$s_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-4}})$	$c_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-5}})$	$p_\psi{}^W_{R\|W}$ $(\mathrm{rad{\cdot}s^{-6}})$
2	4	12	40	155	900

Table 7 gathers the average for all the trajectories of the total time of the trajectory computed by the trajectory planner. The initial and final values, together with the intermediate values at 1%, 5% and 10% of the final value are provided along with their required number of iterations. The columns $\delta\psi$ and $\delta\theta$ gather the average of the 16 planned trajectories per each of the two methods to compute the reduced dimensionality difference between two rotations, the yaw angle difference, $\delta\psi$, and the error-quaternion, $\delta\theta$. The last column shows the difference between the two

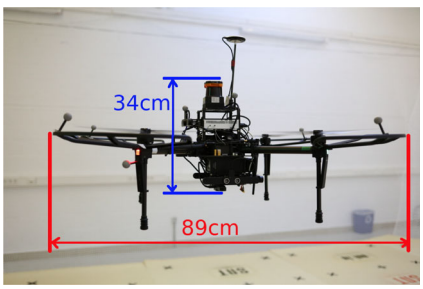


Fig. 7 The aerial robot platform used for the experimental validation

Fig. 8 Our aerial robot flying autonomously inside our flight arena during an experimental validation test. A video with several real flight experiments can be visualized in: https://youtu.be/peR2cTX02Ww

Table 5 Empirically calculated values of the dynamic model of our aerial platform

	x	y	z	ψ
k_j	1.0	1.0	1.0	$\pi/180$
τ_j	0.8355	0.7701	0.5013	0.5142

previous ones. The raw data (before averaging) can be found in the Appendix B, in Table 12 and Fig. 29 for the first path, and in Table 13 and Fig. 30 for the second path.

In all of the cases, except one, the trajectory planner converged to an optimum solution before the maximum allowed number of iterations is reached. In all of the cases, the total time of the trajectory computed by the trajectory planner, using the yaw angle difference, $\delta\psi$, and the error-quaternion, $\delta\theta$, to compute the difference between two rotation, converge to a similar value with a difference lower than 3 %. of improvement with respect to the initial time. Based on these experimental data summarized in Table 7, we can conclude that the error-quaternion, $\delta\theta$, converges to a slightly better optimum than the yaw angle difference, $\delta\psi$, but it requires more iterations.

As it is expected, the optimization process carried out by the trajectory planner eventually converges to a feasible local solution that minimizes the total time of trajectory. It is worth to highlight that in case of real-time constraints and / or limited computational resources availability, the user would be able to limit the maximum number of iterations of the optimization process. In such a case, the calculated trajectory would still be feasible (i.e. all the constraints would be fulfilled), but it would not have converged to a local minimum of the total time of trajectory. Nevertheless, this solution would be a better choice than the initial trajectory. According to the experimental data summarized in Table 7, the error-quaternion, $\delta\theta$, is faster than the yaw angle difference, $\delta\psi$, reaching the 1%, 5% and 10% of the optimum value. Therefore, it is preferable to use the error-quaternion, $\delta\theta$, as the method to compute the reduced dimensionality difference between two rotation, for real-time applications.

Table 8 shows the percentage of the configuration parameters with respect to their the maximum value for the magnitudes of the planned trajectories. The column Init shows the average of the eight initial trajectories (four per each of the two evaluation paths). The first two columns of the column named as Planned gather the average of the 16 planned trajectories per each of the two methods to compute the reduced dimensionality difference between two rotations, the yaw angle difference, $\delta\psi$, and the error-quaternion, $\delta\theta$. The last column shows the difference between the two previous ones. The raw data (before averaging) can be found in the Appendix B, in Tables 14 and 16 for the first path, and in Tables 15 and 17 for the second path.

As it can be extracted from the experiments, the maximum values of the planned trajectories increase with respect to the ones of the initial one, as the optimization process increases them (without over exceeding them) to minimize the total trajectory time. The reader should observe that any of the values of the planned trajectories are higher than 100 %, which means that the maximum values have never been exceeded.

It is desirable that, for the same total trajectory time, the maximum values are the lower possible. The last column of Table 8 can be used to determine which method to compute the reduced dimensionality difference between two rotations is better. A positive number on the last column of the table means that the error-quaternion, $\delta\theta$, generates lower maximum values than the yaw angle difference, $\delta\psi$. According to the experimental data, the error-quaternion, $\delta\theta$, is slightly better than the yaw angle difference, $\delta\psi$.

Table 9 displays the percentage of the energy (computed with Eq. 88) of the planned trajectory magnitudes with respect to the initial trajectories. The columns $\delta\psi$ and $\delta\theta$ gather the average of the 16 planned trajectories per each of the two methods to compute the reduced dimensionality difference between two rotations, the yaw angle difference, $\delta\psi$, and the error-quaternion, $\delta\theta$. The last column shows the difference between the two previous ones. The raw data (before averaging) can be found in the Appendix B, in Table 18 for the first path, and in Table 19 for the second path.

It can be extracted from the experiments, summarized in Table 9, that on average, the total energy is increased with respect to the initial trajectory, as the planned trajectory is more aggressive than the initial one. It is interesting to highlight that the total energy of the higher-order derivatives is reduced with respect to the initial trajectory, while the lower order ones have the opposite behavior. We can, therefore, confirm, based on the experimental data, that our trajectory planner reduces the total energy of the

Table 6 Configuration control commands (i.e. actuator) limits of our aerial platform used in both the trajectory planner and the trajectory tracking controller

	u_x		u_y		u_z		u_ψ	
	Pla.	Ctr.	Pla.	Ctr.	Pla.	Ctr.	Pla.	Ctr.
Min.	−3.0	−4.0	−3.0	−4.0	−3.0	−4.0	−100.0	−100.0
Max.	3.0	4.0	3.0	4.0	3.0	4.0	100.0	100.0

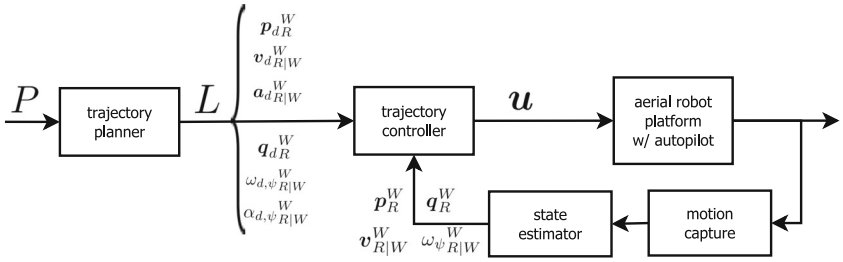
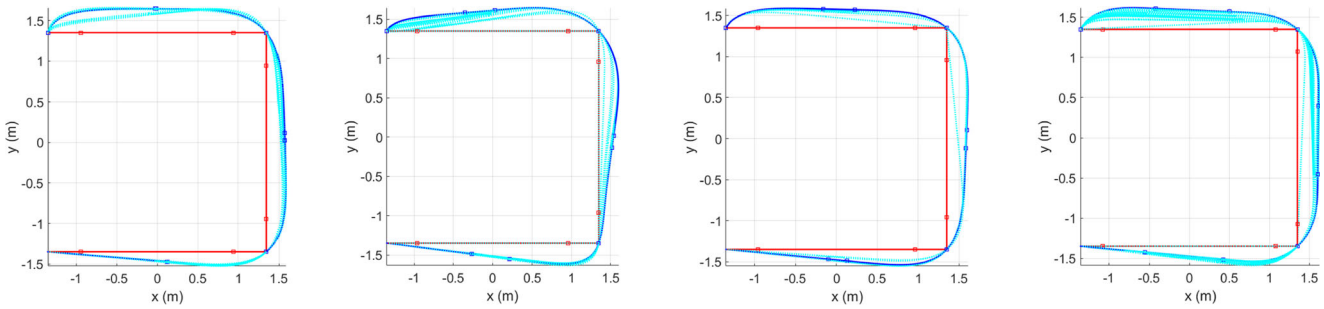


Fig. 9 System architecture setup for the experimental evaluation of the proposed trajectory planner



(a) Slow. Inaccurate (b) Medium-Slow. Inaccurate (c) Medium-Fast. Inaccurate (d) Fast. Inaccurate

Fig. 10 Illustration of the optimization process of the trajectory planner displaying the first 4 waypoints of the first path, configured with an inaccurate maximum linear distance to the path, and using the error-quaternion, $\delta\theta$, to compute the difference between two rotation. In solid red, the initial trajectory that is coincident with the path (in solid black). In dashed cyan, the intermediate trajectories computed at different iterations of the optimization process. In solid blue, the final optimum computed trajectory

Table 7 Total time of the trajectory computed by the trajectory planner, average for all the trajectories

Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi - \delta\theta$
Fin	# it.	**37.4375**	42.3750	−4.9375
	t (s)	18.9438	**18.8494**	0.0944
1 %	# it.	20.4375	**17.4375**	3.0000
	t (s)	19.0694	**18.9694**	0.1000
5 %	# it.	11.2500	**9.0625**	2.1875
	t (s)	**19.4719**	19.4831	−0.0112
10 %	# it.	7.3125	**5.8750**	1.4375
	t (s)	20.4737	**20.1388**	0.3350
Init	# it.	0	0	0
	t (s)	43.0763	43.0763	0
% improv.		−56.0581	**−56.2369**	0.1787

The initial and final values, together with the intermediate values at 1%, 5% and 10% of the final value are provided along with their required number of iterations

Minimum values highlighted in bold

Table 8 Percentage of the configuration parameters with respect to their maximum value for the magnitudes of the planned trajectories

		Init	Planned		
Rot. error			$\delta\psi$	$\delta\theta$	$\delta\psi - \delta\theta$
Max. d		0.0	85.4219	83.7288	1.6931
Vel.	**Lin.**	52.9375	**99.9906**	99.9950	−0.0044
	Ang.	40.6075	**73.2269**	73.5100	−0.2831
Acc.	**Lin.**	39.7900	**99.5219**	99.6637	−0.1419
	Ang.	30.5350	53.1075	**51.5856**	1.5219
Jer.	**Lin.**	32.3912	**69.8181**	70.8212	−1.0031
	Ang.	24.8713	52.0438	**47.4475**	4.5962
Sna.	**Lin.**	90.7938	99.9819	**99.9750**	0.0069
	Ang.	69.7375	89.3981	**85.7806**	3.6175
Cra.	**Lin.**	97.7763	95.5644	**94.0231**	1.5412
	Ang.	71.7413	86.0206	**85.9800**	0.0406
Pop	**Lin.**	30.9637	34.6644	**31.7262**	2.9381
	Ang.	19.0962	**25.8731**	27.7581	−1.8850
Average	**Lin.**	57.4412	83.1937	**82.7219**	0.4719
	Ang.	42.7550	63.1544	**62.0531**	1.1013
	Total	50.0962	73.2369	**72.3250**	0.9119
$\mathbf{u_x}$	**Min.**	40.1887	**84.0950**	85.0294	−0.9344
	Max.	41.7762	86.7287	**86.3931**	0.3356
$\mathbf{u_y}$	**Min.**	31.8000	69.0350	**67.7144**	1.3206
	Max.	26.5700	66.2706	**63.4663**	2.8044
$\mathbf{u_z}$	**Min.**	11.6212	**24.6281**	25.1250	−0.4969
	Max.	6.9437	**12.0381**	13.1019	−1.0637
$\mathbf{u_\psi}$	**Min.**	50.5938	82.6463	**81.7556**	0.8906
	Max.	22.2375	41.2031	**39.3550**	1.8481
Average		28.9650	58.3312	**57.6487**	0.6825
Average		39.6625	68.1400	**67.3719**	0.7681

Average for all the trajectories, differentiating between the two proposed methods to compute the reduced dimensionality difference between two rotations

Minimum values highlighted in bold

higher-order derivatives when minimizing the total trajectory time at the cost of increasing the total energy lower order derivatives. This behavior emerges without been explicitly programmed.

It is desirable that, for the same total trajectory time, the total energy values are the lower possible. The last column of Table 9 can be used to determine which method to compute the reduced dimensionality difference between two rotations is better. A positive number on the last column of the table means that the error-quaternion, $\delta\theta$, generates lower total energy values than the yaw angle difference, $\delta\psi$. According to the experimental data, on average, the yaw angle difference, $\delta\psi$, is slightly better than the error-quaternion, $\delta\theta$. Nevertheless, it is worth to highlight that the error-quaternion, $\delta\theta$, has a better performance in the angular magnitudes than in the linear ones.

Table 10 displays the percentage of the energy (computed with Eq. 88) of the planned control command references with respect to the initial trajectories. The columns $\delta\psi$ and $\delta\theta$ gather the average of the 16 planned trajectories per each of the two methods to compute the reduced dimensionality difference between two rotations, the yaw angle difference, $\delta\psi$, and the error-quaternion, $\delta\theta$. The last column shows the difference between the two previous ones. The raw data (before averaging) can be found in the Appendix B, in Table 20 for the first path, and in Table 21 for the second path.

Similarly than for the case of the trajectory magnitudes, for the case of control command references, it can be extracted from the experiments, summarized in Table 10, that on average, the total energy is increased with respect to the initial trajectory, as the planned trajectory is more aggressive than the initial one.

Table 9 Percentage of the energy (88) of the planned trajectory magnitudes with respect to the initial trajectories

Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi$ - $\delta\theta$
Vel.	**Lin.**	**17.6225**	24.5913	−6.9687
	Ang.	100.7581	**100.7419**	0.0162
Acc.	**Lin.**	**–11.2225**	–6.6869	–4.5356
	Ang.	153.3263	**148.7037**	4.6225
Jer.	**Lin.**	**–31.2831**	–28.6687	–2.6144
	Ang.	**90.3188**	90.9425	–0.6238
Sna.	**Lin.**	**–38.2250**	–36.6337	–1.5913
	Ang.	53.7319	**51.5231**	2.2087
Cra.	**Lin.**	**–35.1731**	–34.0381	–1.1350
	Ang.	–8.7419	**–11.1669**	2.4250
Pop	**Lin.**	**–21.5456**	–20.7713	–0.7744
	Ang.	**–31.2850**	–27.2587	–4.0263
Average	**Lin.**	**–19.9719**	–16.9750	–2.9969
	Ang.	59.6847	**58.9350**	0.7497
	Total	**19.8564**	20.9800	–1.1236

Average for all the trajectories, differentiating between the two proposed methods to compute the reduced dimensionality difference between two rotations

Minimum values highlighted in bold

It is desirable that, for the same total trajectory time, the total energy values are the lower possible. The last column of Table 10 can be used to determine which method to compute the reduced dimensionality difference between two rotations is better. A positive number on the last column of the table means that the error-quaternion, $\delta\theta$, generates lower total energy values than the yaw angle difference, $\delta\psi$. According to the experimental data, on average, the yaw angle difference, $\delta\psi$, is slightly better than the error-quaternion, $\delta\theta$. The reader must note that this difference is especially high in the u_z control command reference.

After the deep analysis of the experimental data presented above, we cannot conclude that there exists a clear dominant method to compute the difference between two rotation. We prefer to use the error-quaternion, $\delta\theta$, over the yaw angle difference, $\delta\psi$, since it is more suitable for real-time applications without a significant difference in performance. In the remainder experimental part of the paper, we use the error-quaternion, $\delta\theta$, to compute the difference between two rotation.

Table 10 Percentage of the energy (88) of the planned control commands references with respect to the initial trajectories

Rot. error	$\delta\psi$	$\delta\theta$	$\delta\psi$ - $\delta\theta$
u_x	**195.5856**	196.5512	−0.9656
u_y	182.3600	**181.0913**	1.2687
u_z	**149.4519**	171.6263	−22.1744
u_ψ	108.4969	**108.2475**	0.2494
Average	**158.9737**	164.3788	−5.4050

Average for all the trajectories, differentiating between the two proposed methods to compute the reduced dimensionality difference between two rotations

Minimum values highlighted in bold

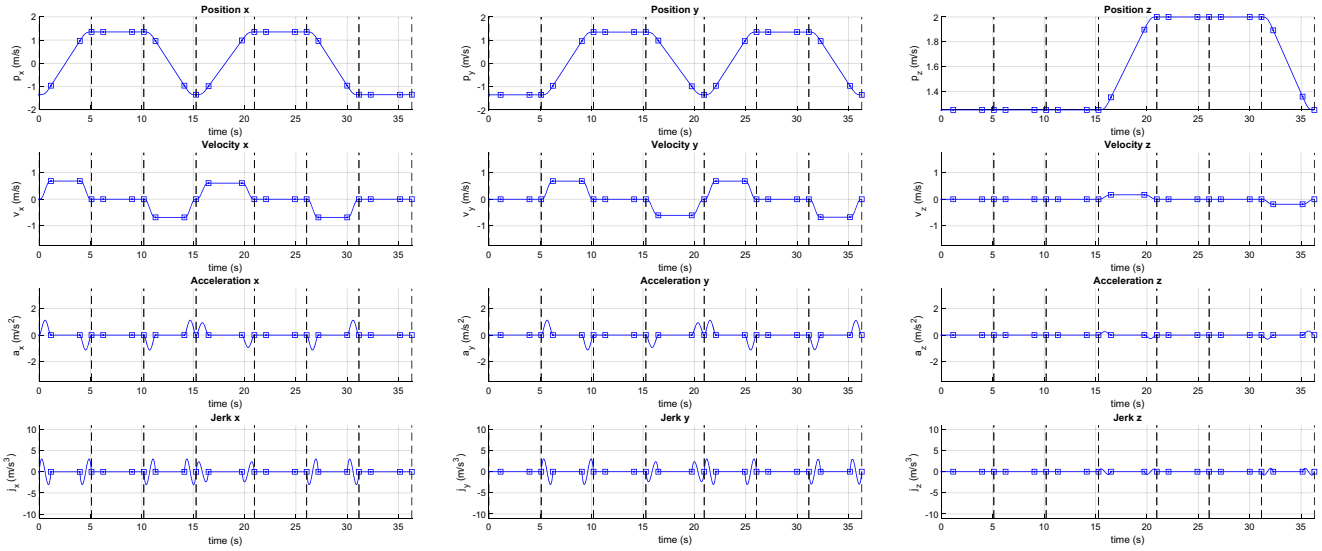
The second part of the trajectory planning results aims to deeply illustrate the trajectory planning process by using two example trajectories.

The first example trajectory has the first path as reference (see Table 1), the inaccurate maximum linear distance to the path, d_{max} (see Table 3) and the Medium-Fast limits on the dynamics of the trajectory (see Table 4). We have set 10 as maximum number of iterations, but the optimization converged at 9 iterations.

The second example trajectory has the second path as reference (see Table 2), the accurate maximum linear distance to the path, d_{max} (see Table 3) and the Medium-Slow limits on the dynamics of the trajectory (see Table 4). We have set 4 as the maximum number of iterations, finishing the optimization process before reaching the optimum value.

Figures 11 and 12 plot the values (pose and derivatives up to jerk) of the initial trajectory used in our trajectory planner (see Section 7.4) for the first example trajectory. The total time of this initial trajectory is $t = 36.29$ s.

Figure 13 shows the 3D view of the calculated trajectory after the optimization process has ended for the first example trajectory. The values (pose and derivatives up to



(a) x. (b) y. (c) z.

Fig. 11 Initial trajectory (first example trajectory). Position, and its derivatives up to jerk, plotted in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 36.29$ s

jerk) of this trajectory are plotted in Figs. 14 and 15. The total time of the planned trajectory is $t = 14.93$ s.

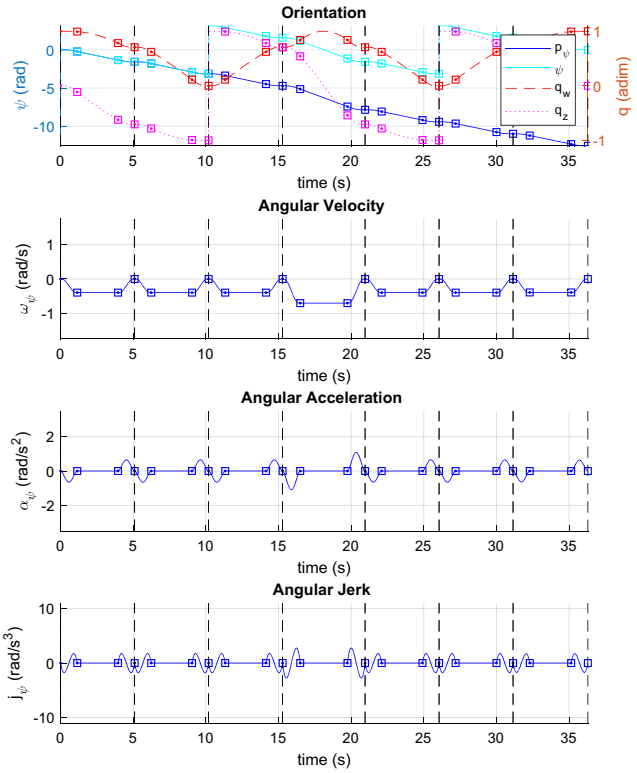


Fig. 12 Initial trajectory (first example trajectory). Heading, and its derivatives up to jerk, plotted in dashed red and magenta and in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 36.29$ s

As expected, the planned trajectory after the optimization process still fulfills all the constraints, but its total trajectory time has decreased by 58.86% from the initial trajectory ($t = 36.29$ s.) to the optimized one ($t = 14.93$ s.). The reader is encourage to deeply compare the values of the initial trajectory, Figs. 11 and 12, with the values of the planned trajectory, Figs. 14 and 15, to perceive the difference between the two trajectories.

Figures 16 and 17 plot the values (pose and derivatives up to jerk) of the initial trajectory used in our trajectory planner for the second example trajectory. The total time of this initial trajectory is $t = 44.36$ s.

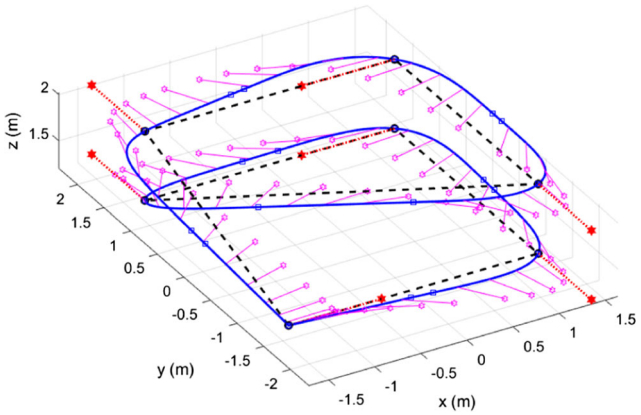
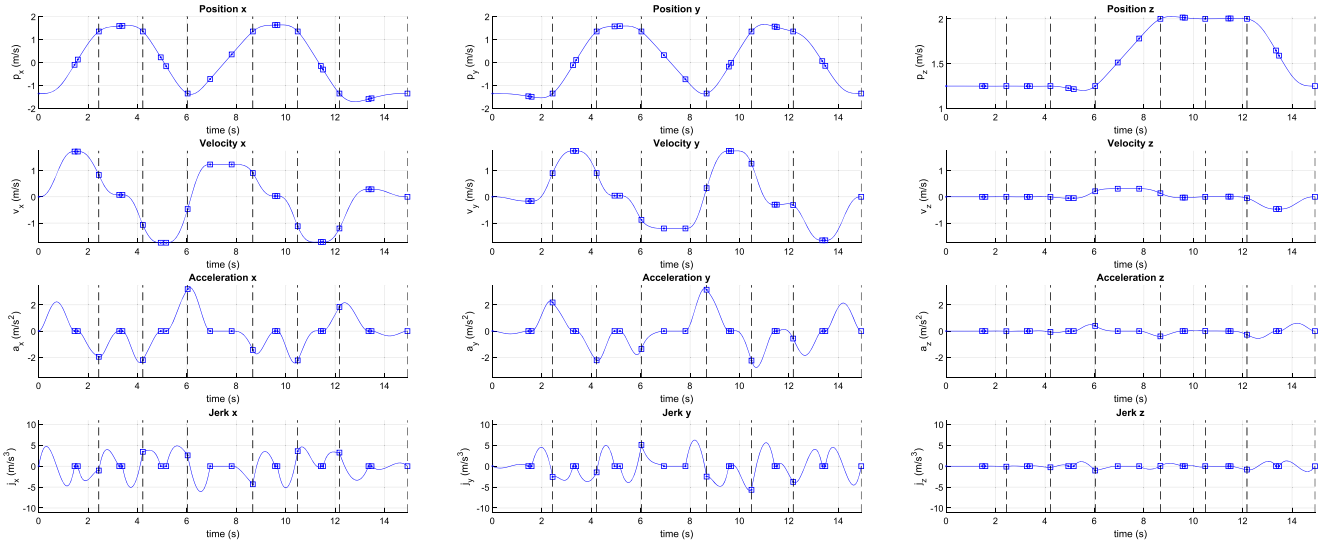


Fig. 13 3D view of the planned trajectory (first example trajectory). The position values are drawn with a solid blue line, whereas the heading is represented with magenta arrows. The path is displayed with a dashed black line, being its waypoints represented by a circle (position) and a red arrow (heading)



(a) x. (b) y. (c) z.

Fig. 14 Planned trajectory (first example trajectory). Position, and its derivatives up to jerk, plotted in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 14.93$ s

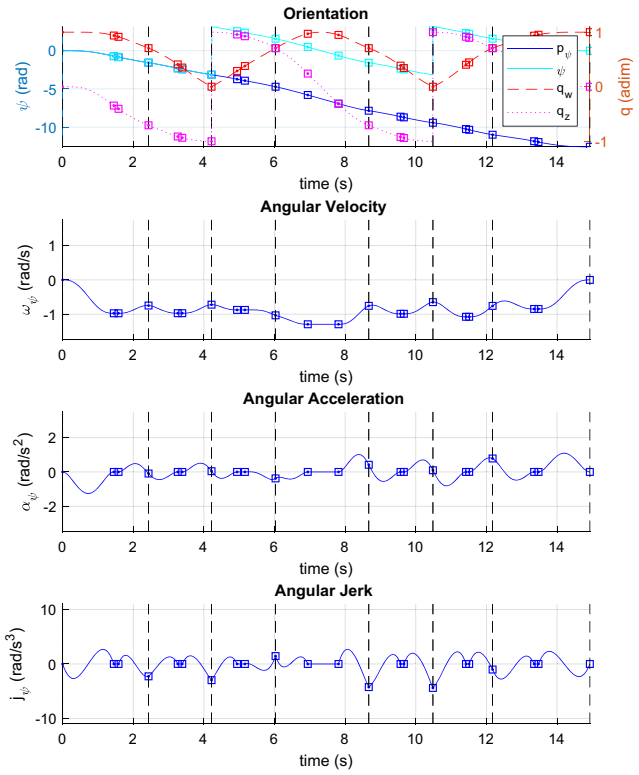


Fig. 15 Planned trajectory (first example trajectory). Heading, and its derivatives up to jerk, plotted in dashed red and magenta and in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 14.93$ s

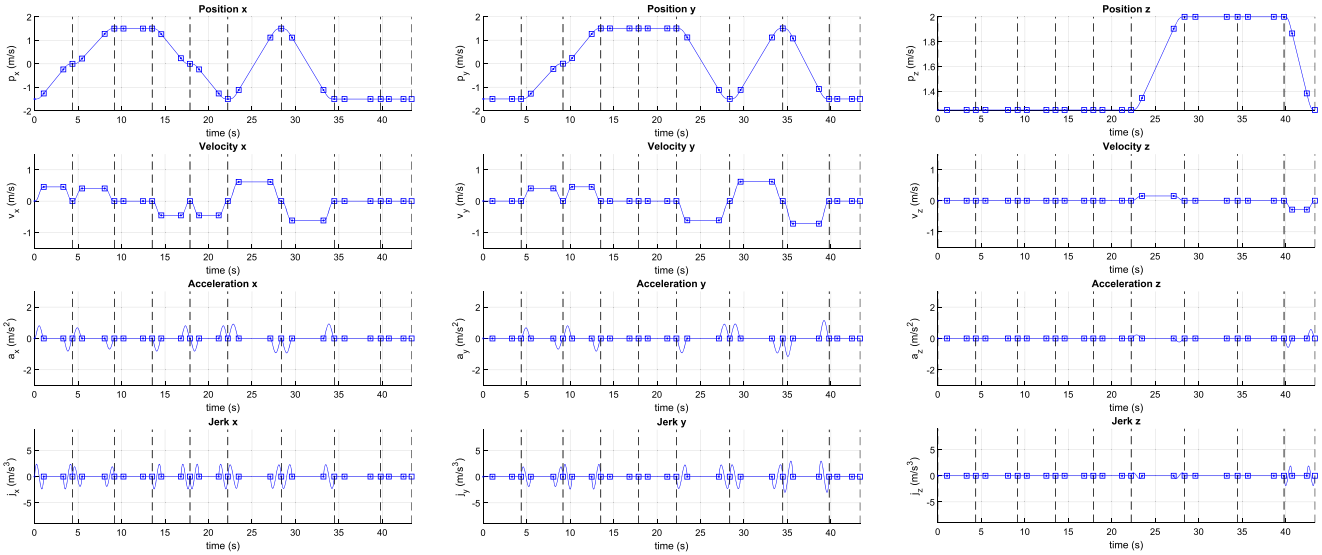
Figure 18 shows the 3D view of the calculated trajectory once the optimization process ended for the second example trajectory.. The values (pose and derivatives up to snap) of this trajectory are plotted in Figs. 19 and 20. The total time of the planned trajectory is $t = 19.46$ s.

As expected, despite forcing to finish the optimization before reaching the optimum value, the planned trajectory after the optimization process finished, still fulfills all the constraints and it total time has decreased by 56.13% from the initial trajectory ($t = 44.36$ s.) to the optimized one ($t = 19.46$ s.). The reader is encourage to deeply compare the values of the initial trajectory, Figs. 16 and 17, with the values of the planned trajectory, Figs. 19 and 20, to perceive the difference between the two trajectories.

9.4 Trajectory Tracking Results

In this section, we illustrate and evaluate the proposed trajectory tracking controller. To analyze the impact of our trajectory planner on the tracking performance, we compare the effect of providing the augmented references (full state and control command) against the traditional pose reference. We have selected 4 of the trajectories presented in Sections 9.1 and 9.3 for real flights with the aerial robot platform presented in Section 9.2.

The estimated pose tracking error is evaluated statistically with the metrics presented in Section 9.1. In Table 11 we compare the tracking performance of the controller



(a) x. (b) y. (c) z.

Fig. 16 Initial trajectory (second example trajectory). Position, and its derivatives up to jerk, plotted in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 44.36$ s

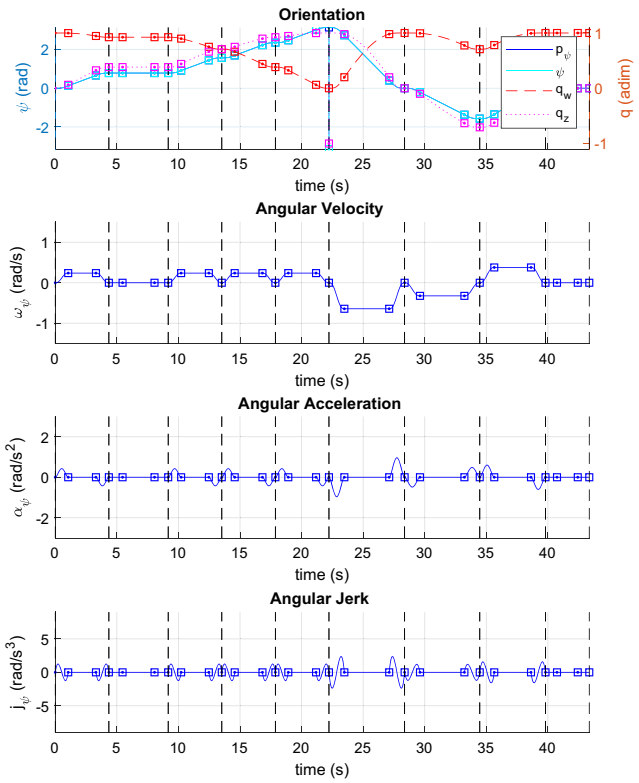


Fig. 17 Initial trajectory (second example trajectory). Heading, and its derivatives up to jerk, plotted in dashed red and magenta and in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 44.36$ s

when providing the augmented reference instead of the pose reference along the 4 selected trajectories. For all experiments, we observe that the controller's tracking performance increases considerably when applying the augmented reference instead of the pose reference. From the experimental data, we can conclude that the additional information (references) generated by the planner actively increases the tracking performance of the controller.

It is worth to mention that the high values of the trajectory tracking error, when compared to another state of the art controllers, is originated from two facts. First,

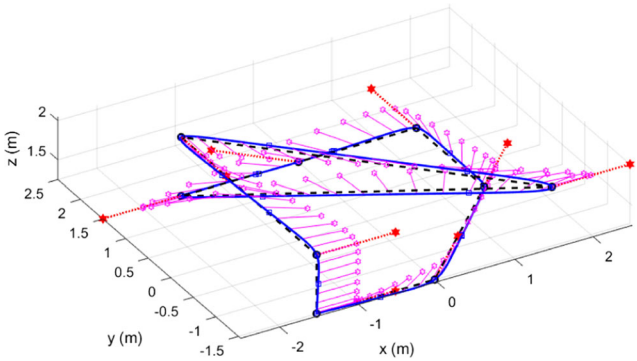


Fig. 18 3D view of the calculated trajectory (second example trajectory). The position values are drawn with a solid blue line, whereas the heading is represented with magenta arrows. The path is displayed with a dashed black line, being its waypoints represented by a circle (position) and a red arrow (heading)

the trajectories are designed with sharp corners and high velocity, which are difficult to track. Second, the modeling errors of the first-order approximation of the robot dynamics, as presented in Section 5. We strongly believe that a more accurate (and complex) model of the aerial robot will drastically reduce the trajectory tracking errors. Nevertheless, as mentioned in Section 5, using such complex models of the aerial robot is out of the scope of the paper.

For illustrative purposes, we have selected two real experiments from Section 9.3 to plot the trajectory tracking performance of the proposed method: (1) First example trajectory: first path, velocity medium-fast, distance inaccurate; (2) Second example trajectory: second path, velocity medium-slow, distance accurate. Different executions of the two example trajectories can be visualized in the video: https://youtu.be/peR2cTX02Ww.

Figure 21 plots the 3D view of the trajectory tracking for the first example trajectory. It presents the configured path, the planned trajectory computed by the planner, and the estimated executed trajectory, using the two different reference configurations for the trajectory controller.

Figures 22 and 23 show the values of the position, orientation and velocity (both linear and angular) involved in the trajectory tracking task for the first example trajectory. These figures plot the planned trajectory computed by the planner, and the estimated executed trajectory, using the two different configurations of the trajectory controller.

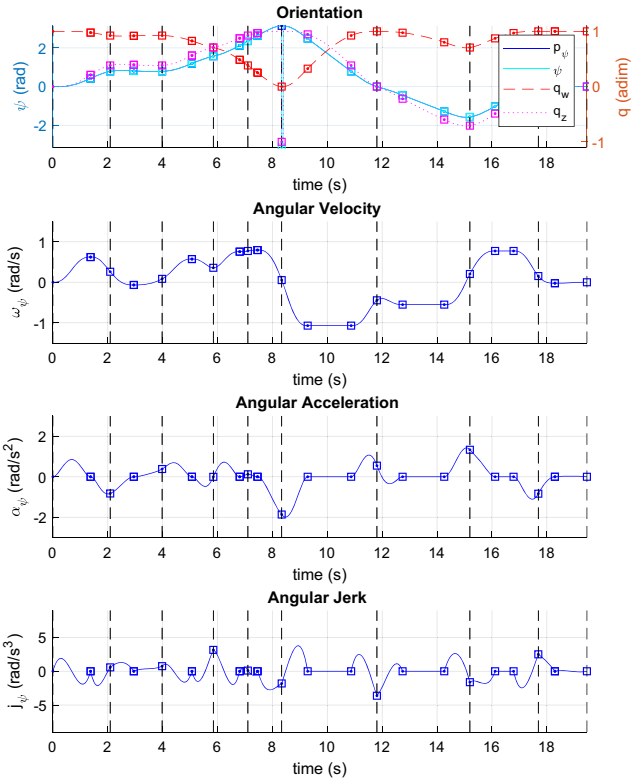
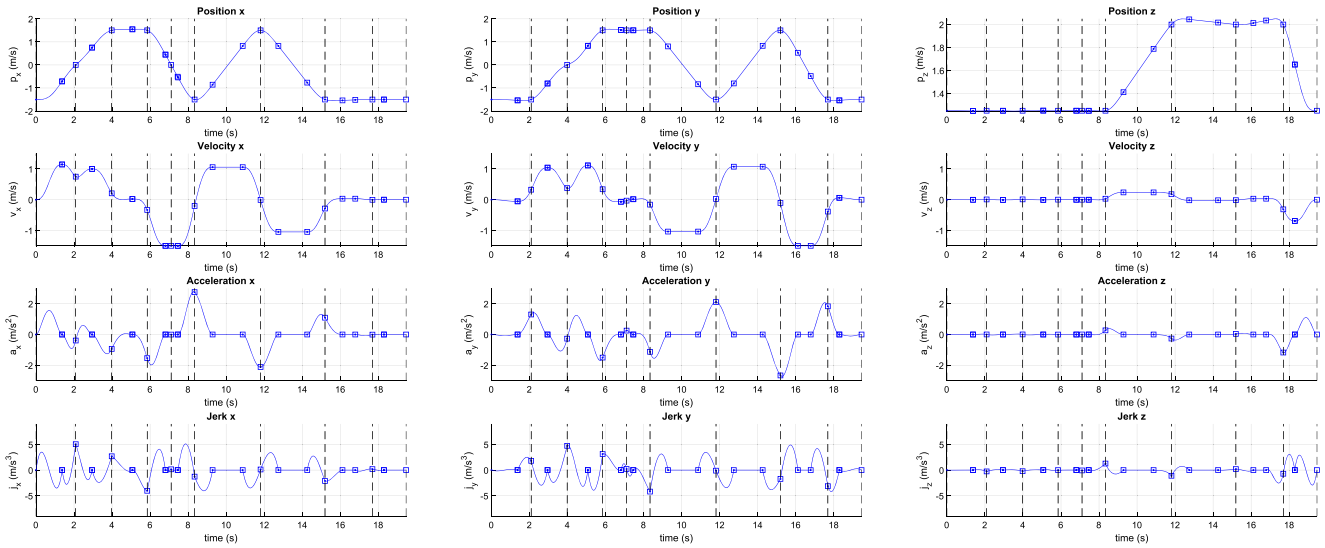


Fig. 20 Calculated trajectory (second example trajectory). Heading, and its derivatives up to jerk, plotted in dashed red and magenta and in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 19.46$ s



(a) x. (b) y. (c) z.

Fig. 19 Calculated trajectory (second example trajectory). Position, and its derivatives up to jerk, plotted in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total trajectory time is $t = 19.46$ s

Figure 24 show the control command references for the first example trajectory. This figure plots the planned control command reference computed by the planner, and the executed control command reference by the controller, using its two different configurations.

Figure 25 plots the 3D view of the trajectory tracking for the second example trajectory. It presents the configured path, the planned trajectory computed by the planner, and the estimated executed trajectory, using the two different configurations of the trajectory controller.

Figures 26 and 27 show the values of the position, orientation and velocity (both linear and angular) involved in the trajectory tracking task for the second example trajectory. These figures plot the planned trajectory computed by the planner, and the estimated executed trajectory, using the two different configurations of the trajectory controller.

Figure 28 show the control command references for the first example trajectory. This figure plots the planned control command reference computed by the planner, and

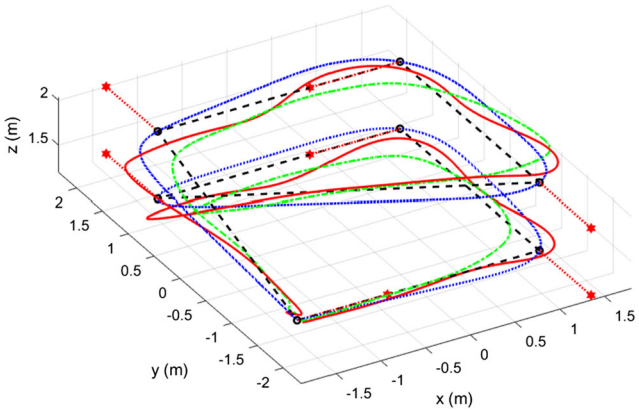
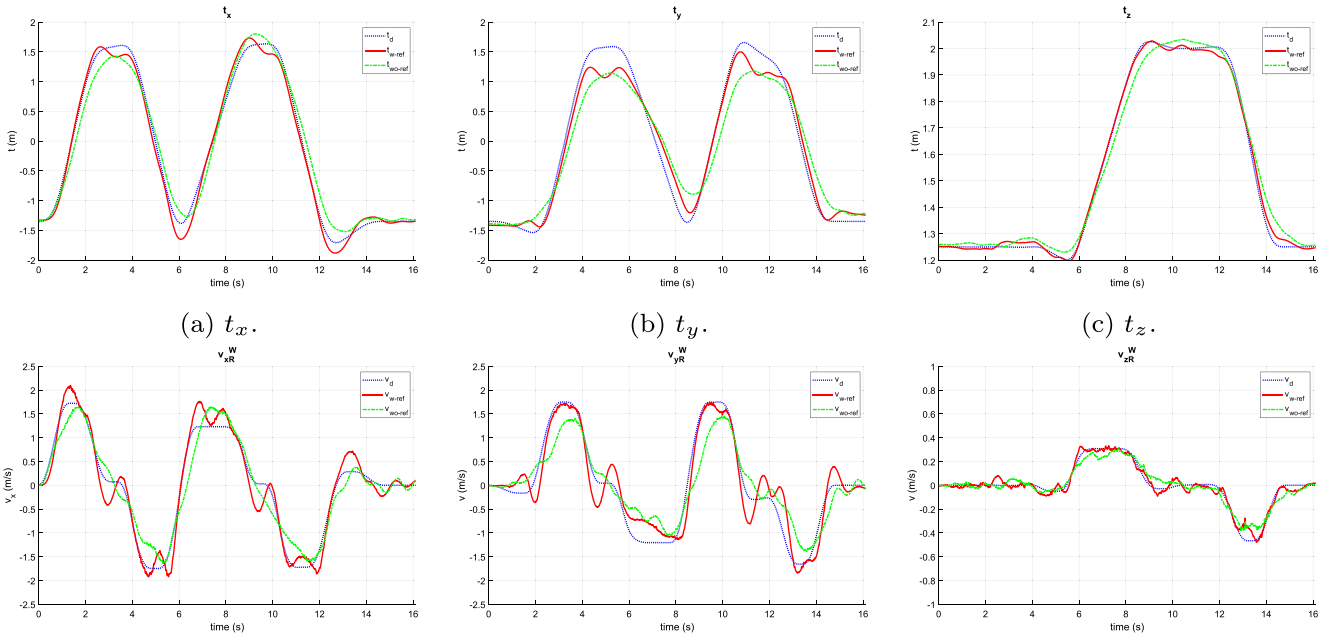


Fig. 21 3D view of the trajectory tracking (first example trajectory). The planned trajectory is represented with a dotted blue line. The estimated executed trajectory with the controller using the full reference is represented with a solid red line. The estimated executed trajectory with the controller using only the pose reference is represented with a dashed green line. The path is displayed with a dashed black line, being its waypoints represented by a circle (position) and a red arrow (heading)

Table 11 Comparison between the planned and the estimated executed trajectories for the four selected trajectories using the metrics presented in Section 9.1, for the two configurations of the trajectory tracking controller: with and without reference

Path		First Path				Second Path			
Velocity		M-F		F		S		M-S	
Distance		Ina		Acc		Acc		Acc	
# it. (Traj. Plan)		9* (10)		10		30		4	
Reference		w/ ref	wo/ ref	w/ ref	wo/ ref	w/ ref	wo/ ref	w/ ref	wo/ ref
δt_x	**MSE**	0.015422	0.031328	0.019823	0.063096	0.0061054	0.033262	0.012565	0.050249
	RMSE	0.12419	0.177	0.14079	0.25119	0.078137	0.18238	0.11209	0.22416
	MAE	0.10423	0.14443	0.10594	0.18916	0.065277	0.15574	0.087567	0.18229
	MaAE	0.26897	0.36199	0.34667	0.50079	0.18314	0.39258	0.30958	0.48297
δt_y	**MSE**	0.043384	0.12033	0.071807	0.20902	0.0088249	0.040129	0.010901	0.066206
	RMSE	0.20829	0.34689	0.26797	0.45718	0.093941	0.20032	0.10441	0.25731
	MAE	0.16852	0.29603	0.2131	0.38576	0.067312	0.16351	0.083794	0.20809
	MaAE	0.51019	0.59049	0.62389	0.84786	0.26408	0.44502	0.22116	0.53269
δt_z	**MSE**	0.00023072	0.0013665	0.00048445	0.001859	0.00016056	0.00088256	0.00042386	0.0019874
	RMSE	0.01519	0.036967	0.02201	0.043116	0.012671	0.029708	0.020588	0.044581
	MAE	0.01174	0.027411	0.018388	0.030916	0.010181	0.018185	0.01525	0.028506
	MaAE	0.036588	0.10118	0.048547	0.13187	0.029855	0.12026	0.065029	0.17138
δt	**MSE**	0.059037	0.15302	0.092114	0.27397	0.015091	0.074274	0.02389	0.11844
	RMSE	0.24298	0.39118	0.3035	0.52342	0.12284	0.27253	0.15456	0.34416
	MAE	0.21559	0.35288	0.2609	0.46822	0.10658	0.25601	0.13658	0.31635
	MaAE	0.53832	0.62114	0.63117	0.90611	0.26687	0.4654	0.3125	0.55515
δ_ψ	**MSE**	0.002323	0.023846	0.0079	0.059133	0.0023006	0.011252	0.0018696	0.019394
	RMSE	0.048197	0.15442	0.088882	0.24317	0.047964	0.10608	0.043239	0.13926
	MAE	0.038429	0.13726	0.069809	0.22115	0.038009	0.087457	0.035343	0.10996
	MaAE	0.12029	0.26095	0.24615	0.43667	0.11551	0.22884	0.092931	0.30698

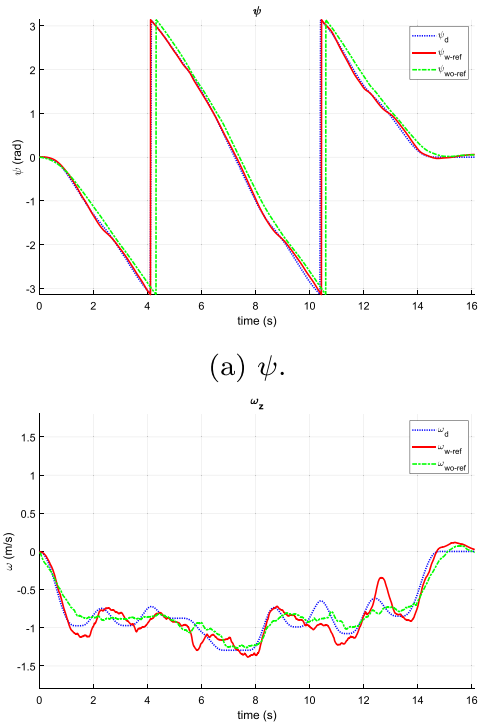
Minimum values highlighted in bold



(a) t_x. (b) t_y. (c) t_z.

(d) v_x. (e) v_y. (f) v_z.

Fig. 22 Position and linear velocity of trajectory tracking (first example trajectory). The planned trajectory is represented with a dotted blue line. The estimated executed trajectory with the controller using the full reference is represented with a solid red line. The estimated executed trajectory with the controller using only the pose reference is represented with a dashed green line



(a) ψ.

(b) ω_ψ.

Fig. 23 Heading (angle) and angular velocity of trajectory tracking (first example trajectory). The planned trajectory is represented with a dotted blue line. The estimated executed trajectory with the controller using the full reference is represented with a solid red line. The estimated executed trajectory with the controller using only the pose reference is represented with a dashed green line

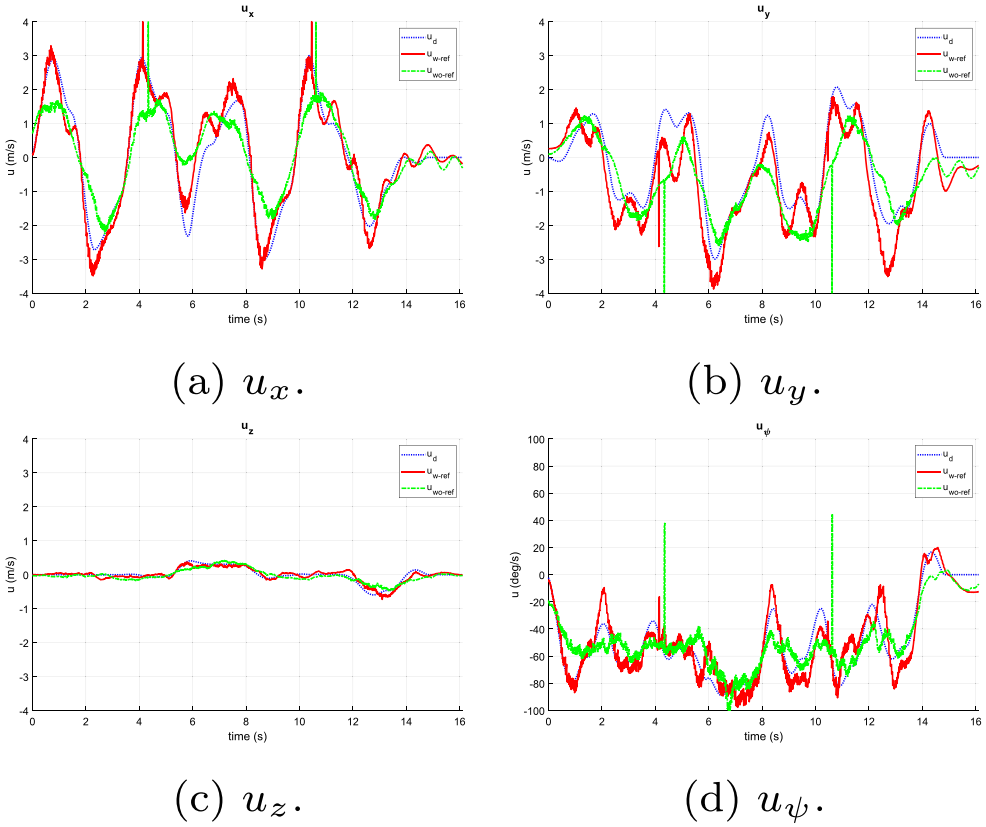
the executed control command reference by the controller, using its two different configurations.

From the experimental plots and data (Table 11) we conclude that, for a given control approach and robot model, the tracking performance can be considerably increased by augmenting the reference information provided from the planning step. Then, the controller finds the optimal trade-off between the different references: pose, velocities, control commands, etc.

10 Conclusions and Future Work

In this paper, we have presented an optimization-based trajectory tracking solution for multirotor aerial robots given a geometrically feasible path, as the continuation of our previous work [50].

Our trajectory planner minimizes the trajectory time and includes not only restrictions such as continuity of the trajectory (i.e. class C^m), limits on velocity, and higher-order derivatives, constraints in the waypoints, and maximum distance between the planned trajectory and the given path, but also restrictions in the actuators of the aerial robot based on its dynamic model, guaranteeing that the planned trajectory will be achievable by the robot. We proposed a novel compact multi-phase trajectory definition as a set of two different kinds of polynomials, providing a higher semantic encoding of the trajectory, which allows



(a) u_x. (b) u_y. (c) u_z. (d) u_ψ.

Fig. 24 Control command references (first example trajectory). The planned control command reference is represented with a dotted blue line. The executed control command reference by the controller using the full reference is represented with a solid red line. The executed control command reference by the controller using only the pose reference is represented with a dashed green line

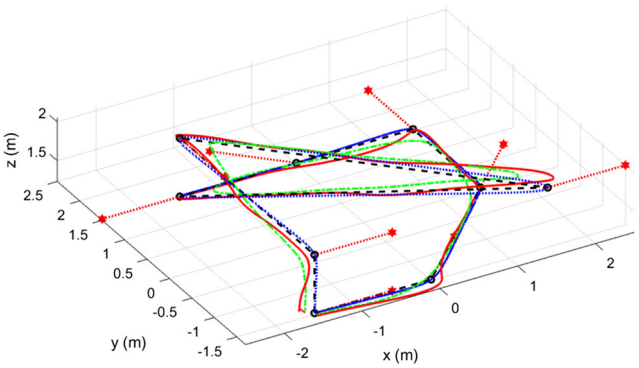
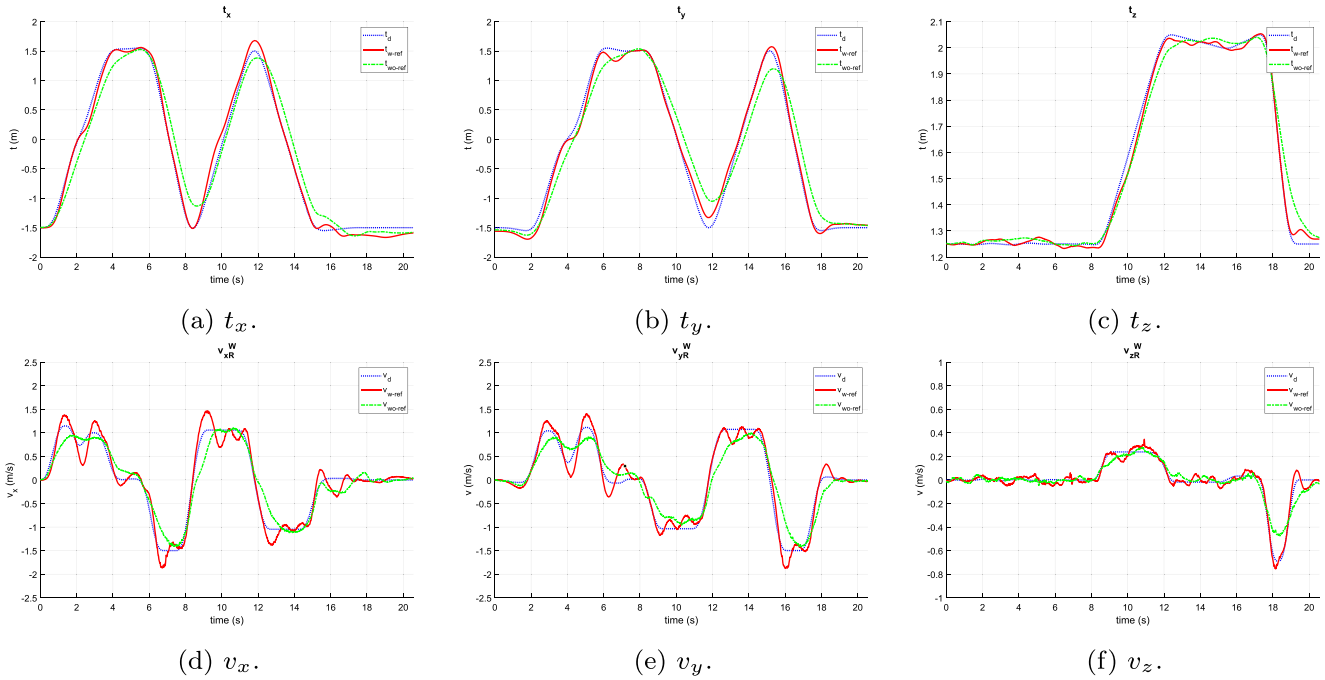


Fig. 25 3D view of the trajectory tracking (second example trajectory). The planned trajectory is represented with a dotted blue line. The estimated executed trajectory with the controller using the full reference is represented with a solid red line. The estimated executed trajectory with the controller using only the pose reference is represented with a dashed green line. The path is displayed with a dashed black line, being its waypoints represented by a circle (position) and a red arrow (heading)

calculating, without loss of flexibility, an optimal solution but following a predefined simple profile.

Our Model Predictive Controller (MPC) for trajectory tracking has been formulated to take as inputs all the magnitudes of the planned trajectory (i.e. position and heading, velocity, and acceleration) as well as the control command references computed using the planned trajectory, to generate the control commands, improving the tracking performance when compared with a controller that only uses the planned position and heading.

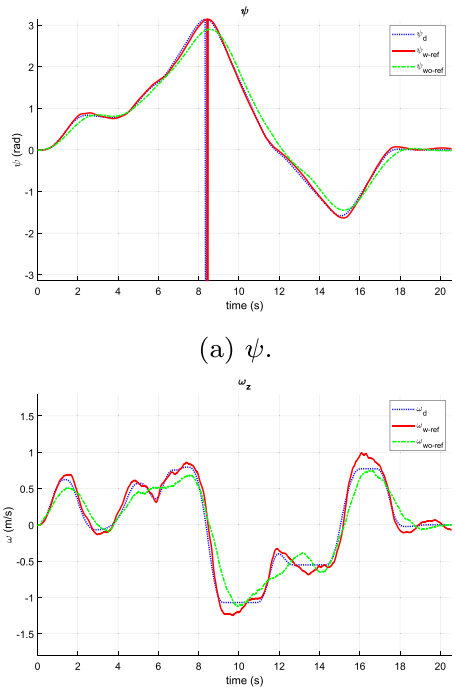
Both planner and controller quaternions and error-quaternions to mathematical represent orientations, as it has been shown that have some advantages when compared to other formulations. To arrive to this conclusion, we have analyzed the most commonly used mathematical representations of orientations and their kinematic relationships in the tridimensional space $SO(3)$ and in the bidimensional space $SO(2)$, together with the computations of the difference between two rotations as



(a) t_x. (b) t_y. (c) t_z.

(d) v_x. (e) v_y. (f) v_z.

Fig. 26 Position and linear velocity of trajectory tracking (second example trajectory). The planned trajectory is represented with a dotted blue line. The estimated executed trajectory with the controller using the full reference is represented with a solid red line. The estimated executed trajectory with the controller using only the pose reference is represented with a dashed green line



(a) ψ.

(b) ω_ψ.

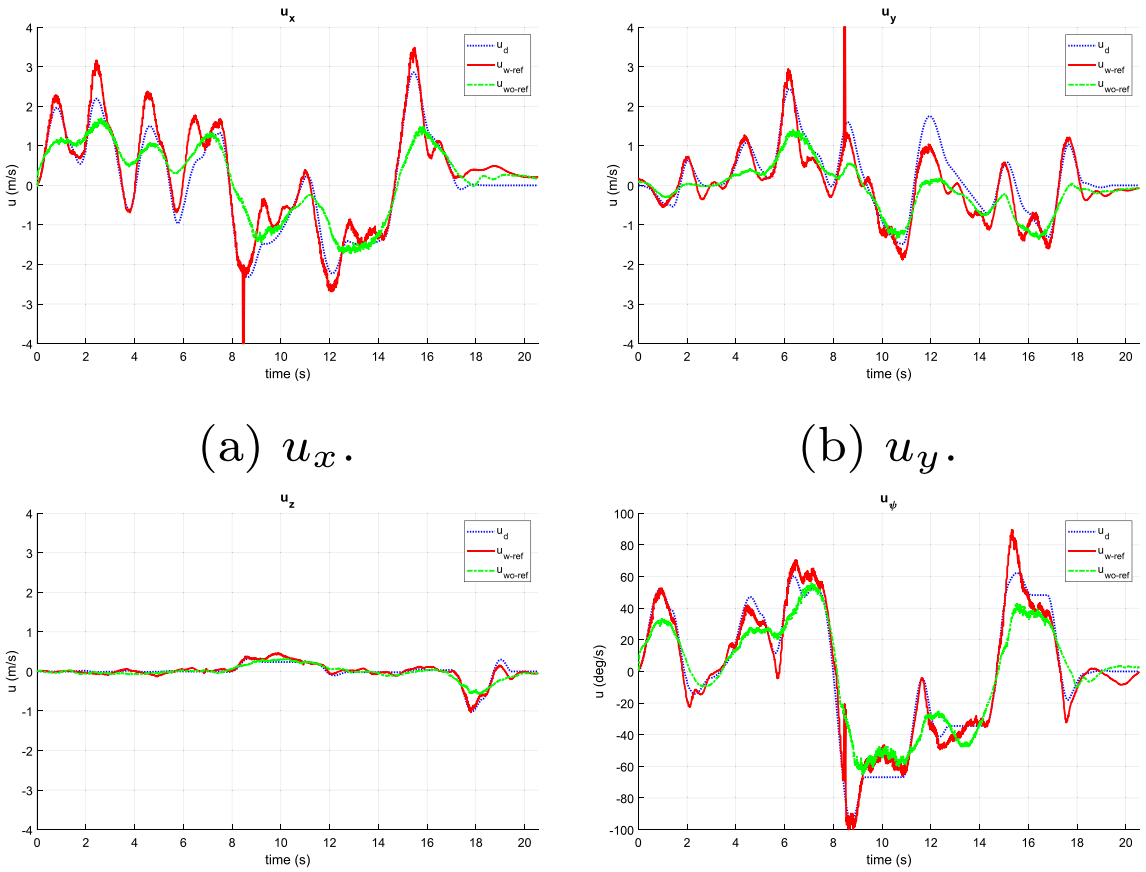
Fig. 27 Heading (angle) and angular velocity of trajectory tracking (second example trajectory). The planned trajectory is represented with a dotted blue line. The estimated executed trajectory with the controller using the full reference is represented with a solid red line. The estimated executed trajectory with the controller using only the pose reference is represented with a dashed green line

well as the definitions of a scalar error between two orientations.

We have validated our trajectory tracking approach thanks to an extensive qualitative and quantitative evaluation. On the one hand, we analyzed the proposed trajectory planner, including its optimization process as well as the properties of the calculated trajectories. On the other hand, we studied the proposed trajectory tracking controller through real flights, comparing its performance with a formulation that takes advantage of the whole output of the trajectory planner (i.e. pose and higher-order derivatives, and control command references) and when only considering the desired pose.

Some future work lines cover the usage, in both the planner and controller, of a more complex robot model to improve the tracking performance. Specifically related to the trajectory planner include its online execution, a multi-objective optimization (for example minimizing time and snap simultaneously), and the incorporation of constraints in the derivatives of linear and angular magnitudes in the waypoints (e.g. velocity in the waypoints).

The trajectory controller will be extended in the future to include a full dynamic model of the robot with $SO(3)$ attitude control. In addition, online model estimation and disturbance rejection are supplementary properties that could be developed to increase tracking performance.



(a) u_x. (b) u_y. (c) u_z. (d) u_ψ.

Fig. 28 Control command references (second example trajectory). The planned control command reference is represented with a dotted blue line. The executed control command reference by the controller using the full reference is represented with a solid red line. The executed control command reference by the controller using only the pose reference is represented with a dashed green line

Author Contributions J.L.S.-L. representation of rotations, trajectory definition, trajectory planner, evaluation and manuscript writing. M.C.-L. trajectory tracking controller, evaluation and manuscript writing. M.A.O.-M. and H.V. project management and funding acquisition.

Funding Information This work was supported by the "Fonds National de la Recherche" (FNR), Luxembourg, under the projects C15/15/10484117 (BEST-RPAS) and PoC16/11565377 (AFI).

Appendix A: Appendix to Trajectory Definition

To ease the formulation of the trajectory planner presented in Section 7, the definition of the trajectory can be expressed using the compact formulation presented along with this Appendix A.

A.1 Position and Derivatives of a Polynomial

The position of a polynomial segment defined in Eq. 67, can be written using the following compact nomenclature:

$$\begin{aligned} p_{i,j}(\tau_i) &= \boldsymbol{l}_{\boldsymbol{x},m_i,0}(\tau_i) \cdot \boldsymbol{b}_{i,j,:} \\ &= \begin{bmatrix} 1 & \tau_i & \tau_i^2 & \dots & \tau_i^{m_i} \end{bmatrix} \cdot \begin{bmatrix} b_{i,j,0} \\ b_{i,j,1} \\ b_{i,j,2} \\ \dots \\ b_{i,j,m_i} \end{bmatrix} \end{aligned} \tag{89}$$

$$\begin{aligned} p_{i,j}(\tau_i) &= (\boldsymbol{b}_{i,j,:})^T \cdot \boldsymbol{r}_{\boldsymbol{x},m_i,0}(\tau_i) \\ &= \begin{bmatrix} b_{i,j,0} & b_{i,j,1} & b_{i,j,2} & \dots & b_{i,j,m_i} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \tau_i \\ \tau_i^2 \\ \dots \\ \tau_i^{m_i} \end{bmatrix} \end{aligned} \tag{90}$$

The l_m-th time-derivative of the position defined in Eq. 68, can be written as:

$$p_{i,j}^{(l_m)}(\tau_i) = \frac{\mathrm{d}^{l_m} p_{i,j}}{\mathrm{d}\tau_i^{l_m}} = \boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i) \cdot \boldsymbol{b}_{i,j,:} \tag{91}$$

$$= (\boldsymbol{b}_{i,j,:})^T \cdot \boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i) \tag{92}$$

where the k-th element of the matrices:

$$\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)_{1,k} = \boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i)_{k,1} = \left(\prod_{l=k-(l_m-1)}^{k} l \right) \cdot \tau_i^{k-l_m} \tag{93}$$

The following equivalence can be easily extracted:

$$\boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i) = \left(\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)\right)^T \tag{94}$$

Additionally, it can be easily demonstrated that:

$$\frac{\mathrm{d}\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)}{\mathrm{d}\tau_i} = \boldsymbol{l}_{\boldsymbol{x},m_i,l_{m+1}}(\tau_i) \tag{95}$$

$$\frac{\mathrm{d}\boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i)}{\mathrm{d}\tau_i} = \boldsymbol{r}_{\boldsymbol{x},m_i,l_{m+1}}(\tau_i) \tag{96}$$

The norm of the magnitude, $p_{i,j}^{(l_m)}(\tau_i)$, can be calculated as:

$$\begin{aligned} \| p_{i,j}^{(l_m)}(\tau_i) \|_2^2 &= \left(p_{i,j}^{(l_m)}(\tau_i)\right)^T \cdot p_{i,j}^{(l_m)}(\tau_i) \\ &= \left(\boldsymbol{b}_{i,j,:}\right)^T \cdot \boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i) \cdot \boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i) \cdot \boldsymbol{b}_{i,j,:} \\ &= \left(\boldsymbol{b}_{i,j,:}\right)^T \cdot \boldsymbol{rl}_{\boldsymbol{x},m_i,l_m}(\tau_i) \cdot \boldsymbol{b}_{i,j,:} \end{aligned} \tag{97}$$

A.2 State of a Polynomial

The state $\boldsymbol{x}_{i,j,l_0:l_m}$ of a polynomial can be defined as:

$$\boldsymbol{x}_{i,j,l_0:l_m}(\tau_i) = \begin{bmatrix} p_{i,j}^{(l_0)} & p_{i,j}^{(l_1)} & \dots & p_{i,j}^{(l_m)} \end{bmatrix}^T \tag{98}$$

Which is calculated using the following compact nomenclature:

$$\boldsymbol{x}_{i,j,l_0:l_m}(\tau_i) = \boldsymbol{L}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i) \cdot \boldsymbol{b}_{i,j,:} \tag{99}$$

$$\left(\boldsymbol{x}_{i,j,l_0:l_m}(\tau_i)\right)^T = (\boldsymbol{b}_{i,j,:})^T \cdot \boldsymbol{R}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i) \tag{100}$$

where

$$\boldsymbol{L}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i) = \begin{bmatrix} \boldsymbol{l}_{\boldsymbol{x},m_i,l_0}(\tau_i) \\ \boldsymbol{l}_{\boldsymbol{x},m_i,l_1}(\tau_i) \\ \dots \\ \boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i) \end{bmatrix} \tag{101}$$

$$\boldsymbol{R}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i) = \left(\boldsymbol{L}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i)\right)^T \tag{102}$$

In the same way than in Eqs. 95 and 96, it can be easily demonstrated that:

$$\frac{\mathrm{d}\boldsymbol{L}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i)}{\mathrm{d}\tau_i} = \boldsymbol{L}_{\boldsymbol{x},m_i,l_1:l_{m+1}}(\tau_i) \tag{103}$$

$$\frac{\mathrm{d}\boldsymbol{R}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i)}{\mathrm{d}\tau_i} = \boldsymbol{R}_{\boldsymbol{x},m_i,l_1:l_{m+1}}(\tau_i) \tag{104}$$

A.3 Coefficients of a polynomial

Given the state of a polynomial, $\boldsymbol{x}_{i,j,l_0:l_m}$, at a certain time τ_i, the coefficients of the polynomial, $\boldsymbol{b}_{i,j,:}$, can be calculated using Eq. 100, as:

$$\boldsymbol{b}_{i,j,:} = \left(\boldsymbol{L}_{\boldsymbol{x},m_i,l_0:l_m}(\tau_i)\right)^{-1} \cdot \boldsymbol{x}_{i,j,l_0:l_m}(\tau_i) \tag{105}$$

with the condition that $l_0 = 0$ and $l_m = m_i$.

The coefficients of a polynomial, $\boldsymbol{b}_{i,j,:}$, can be calculated from its initial state, $\boldsymbol{x}_{i,j,0:m_i}(0)$, using (105).

$$\boldsymbol{b}_{i,j,:} = \left(\boldsymbol{L}_{\boldsymbol{x},m_i,0:m_i}(0)\right)^{-1} \cdot \boldsymbol{x}_{i,j,0:m_i}(0) \tag{106}$$

being therefore the value of the coefficients, independent of the time, $\Delta\tau_i$.

A.4 Initial and Final States of a Polynomial

The initial state of a polynomial, $\boldsymbol{x}_{i,j,0:m_i}(0)$, can be calculated following (100), as:

$$\boldsymbol{x}_{i,j,0:m_i}(0) = \boldsymbol{L}_{\boldsymbol{x},m_i,0:m_i}(0) \cdot \boldsymbol{b}_{i,j,:} \tag{107}$$

The final state of a polynomial, $\boldsymbol{x}_{i,j,0:m_i}(\Delta\tau_i)$, can be calculated from its initial state as follows:

$$\begin{aligned} \boldsymbol{x}_{i,j,0:m_i}(\Delta\tau_i) &= \boldsymbol{L}_{\boldsymbol{x},m_i,0:m_i}(\Delta\tau_i) \cdot \boldsymbol{b}_{i,j,:} \\ &= \boldsymbol{L}_{\boldsymbol{x},m_i,0:m_i}(\Delta\tau_i) \cdot \left(\boldsymbol{L}_{\boldsymbol{x},m_i,0:m_i}(0)\right)^{-1} \cdot \boldsymbol{x}_{i,j,0:m_i}(0) \end{aligned} \tag{108}$$

A.5 Linear Variables of a Segment

The full-dimensional position of the segment is defined as:

$$\boldsymbol{p}_{i,:}(\tau_i) = \begin{bmatrix} p_{i,x}(\tau_i) & p_{i,y}(\tau_i) & p_{i,z}(\tau_i) \end{bmatrix}^T \tag{109}$$

Which is calculated using the following compact nomenclature:

$$\boldsymbol{p}_{i,:}(\tau_i) = \underline{\boldsymbol{l}_{\boldsymbol{x},m_i,0}}(\tau_i) \cdot \underline{\boldsymbol{b}_{i,xyz,:}} \tag{110}$$

$$\left(\boldsymbol{p}_{i,:}(\tau_i)\right)^T = \left(\underline{\boldsymbol{b}_{i,xyz,:}}\right)^T \cdot \underline{\boldsymbol{r}_{\boldsymbol{x},m_i,0}}(\tau_i) \tag{111}$$

where

$$\underline{\boldsymbol{b}_{i,xyz,:}} = \begin{bmatrix} \boldsymbol{b}_{i,x,:} \\ \boldsymbol{b}_{i,y,:} \\ \boldsymbol{b}_{i,z,:} \end{bmatrix} \tag{112}$$

$$\underline{\boldsymbol{l}_{\boldsymbol{x},m_i,0}(\tau_i)} = \begin{bmatrix} \boldsymbol{l}_{\boldsymbol{x},m_i,0}(\tau_i) & \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{0}_{1\times(m_i+1)} \\ \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{l}_{\boldsymbol{x},m_i,0}(\tau_i) & \boldsymbol{0}_{1\times(m_i+1)} \\ \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{l}_{\boldsymbol{x},m_i,0}(\tau_i) \end{bmatrix} \tag{113}$$

$$\underline{\boldsymbol{r}_{\boldsymbol{x},m_i,0}(\tau_i)} = \left(\underline{\boldsymbol{l}_{\boldsymbol{x},m_i,0}(\tau_i)}\right)^T \tag{114}$$

The l_m-th time-derivative of the position can be expressed as:

$$\boldsymbol{p}_{i,:}^{(l_m)}(\tau_i) = \frac{\mathrm{d}^{l_m}\boldsymbol{p}_{i,:}}{\mathrm{d}\tau_i^{l_m}} = \begin{bmatrix} p_{i,x}^{(l_m)}(\tau_i) & p_{i,y}^{(l_m)}(\tau_i) & p_{i,z}^{(l_m)}(\tau_i) \end{bmatrix}^T \tag{115}$$

Which is calculated using the following compact nomenclature:

$$\boldsymbol{p}_{i,:}^{(l_m)}(\tau_i) = \underline{\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)} \cdot \underline{\boldsymbol{b}_{i,xyz,:}} \tag{116}$$

$$\left(\boldsymbol{p}_{i,:}^{(l_m)}(\tau_i)\right)^T = \left(\underline{\boldsymbol{b}_{i,xyz,:}}\right)^T \cdot \underline{\boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i)} \tag{117}$$

where

$$\underline{\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)} = \begin{bmatrix} \boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i) & \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{0}_{1\times(m_i+1)} \\ \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i) & \boldsymbol{0}_{1\times(m_i+1)} \\ \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{0}_{1\times(m_i+1)} & \boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i) \end{bmatrix} \tag{118}$$

$$\underline{\boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i)} = \left(\underline{\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)}\right)^T \tag{119}$$

It can be easily demonstrated that:

$$\frac{\mathrm{d}\underline{\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)}}{\mathrm{d}\tau_i} = \underline{\boldsymbol{l}_{\boldsymbol{x},m_i,l_{m+1}}(\tau_i)} \tag{120}$$

$$\frac{\mathrm{d}\underline{\boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i)}}{\mathrm{d}\tau_i} = \underline{\boldsymbol{r}_{\boldsymbol{x},m_i,l_{m+1}}(\tau_i)} \tag{121}$$

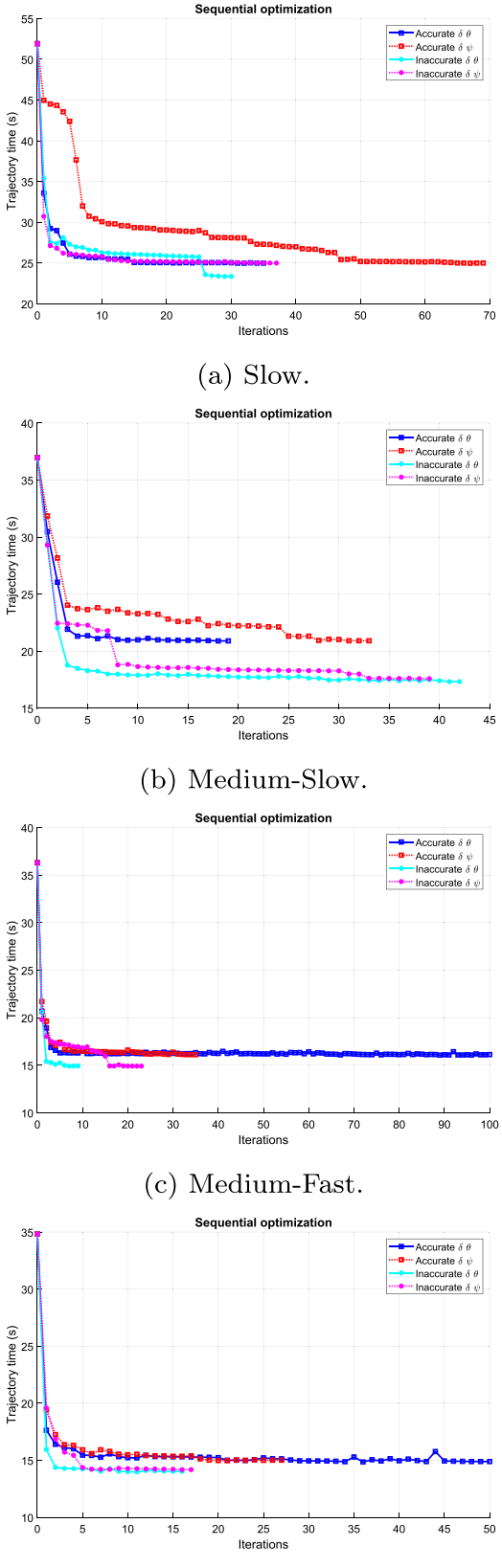
The norm of the magnitude, $\boldsymbol{p}_{i,:}^{(l_m)}(\tau_i)$, can be calculated as:

$$\begin{aligned} \|\boldsymbol{p}_{i,:}^{(l_m)}(\tau_i)\|_2^2 &= \left(\boldsymbol{p}_{i,:}^{(l_m)}(\tau_i)\right)^T \cdot \boldsymbol{p}_{i,:}^{(l_m)}(\tau_i) \\ &= \left(\underline{\boldsymbol{b}_{i,xyz,:}}\right)^T \cdot \underline{\boldsymbol{r}_{\boldsymbol{x},m_i,l_m}(\tau_i)} \cdot \underline{\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)} \cdot \underline{\boldsymbol{b}_{i,xyz,:}} \\ &= \left(\underline{\boldsymbol{b}_{i,xyz,:}}\right)^T \cdot \underline{\boldsymbol{r}\boldsymbol{l}_{\boldsymbol{x},m_i,l_m}(\tau_i)} \cdot \underline{\boldsymbol{b}_{i,xyz,:}} \end{aligned} \tag{122}$$

Appendix B: Appendix to Evaluation and Results

This Appendix B gathers extra information and the raw data of the experiments presented in Section 9.

Figures 29 and 30 graphically represent the total time of the trajectory tracking computed by the trajectory planner



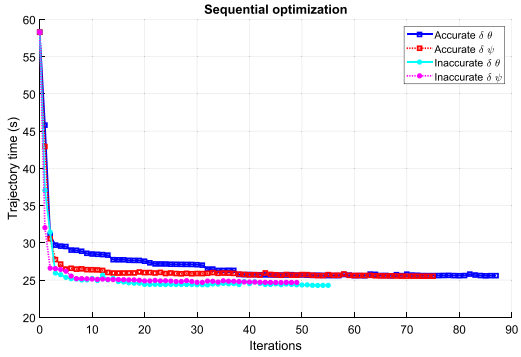
(a) Slow.

(b) Medium-Slow.

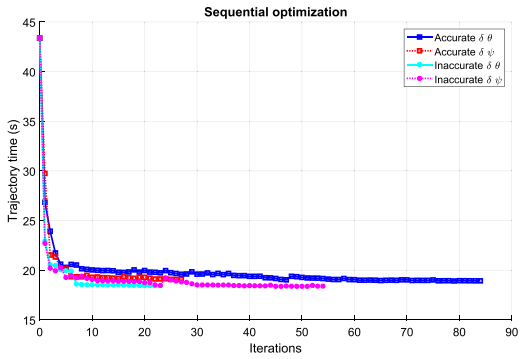
(c) Medium-Fast.

(d) Fast.

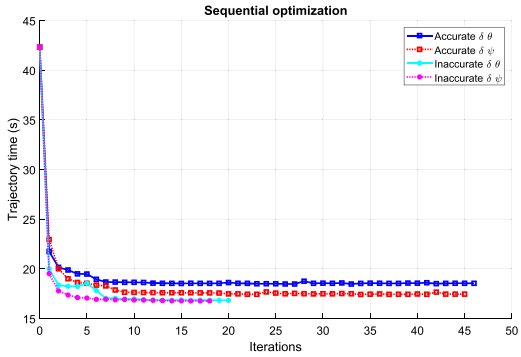
Fig. 29 Total time of the trajectory tracking computed by the trajectory planner during the optimization process for the first evaluation path



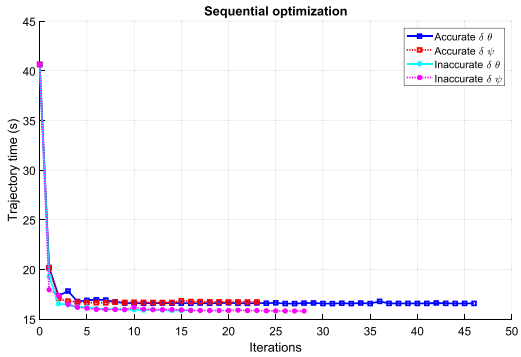
(a) Slow.



(b) Medium-Slow.



(c) Medium-Fast.



(d) Fast.

Fig. 30 Total time of the trajectory tracking computed by the trajectory planner during the optimization process for the second evaluation path

Table 12 Total time of the trajectory tracking computed by the trajectory planner for the first evaluation path

Velocity		**S**				**M-S**				**M-F**				**F**			
Distance		**Acc**		**Ina**		**Acc**		**Ina**		**Acc**		**Ina**		**Acc**		**Ina**	
Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
Fin	# it.	69	35	37	30	33	19	39	42	35	100*	23	9	27	50	17	16
	t (s)	25.00	24.98	24.99	23.35	20.91	20.90	17.60	17.33	16.11	16.12	14.91	14.93	15.00	14.89	14.18	14.04
1 %	# it.	51	16	15	27	29	7	34	30	25	12	17	7	19	22	7	8
	t (s)	25.19	25.06	25.23	23.57	20.94	21.10	17.63	17.48	16.20	16.23	14.92	14.99	15.11	15.00	14.22	14.04
5 %	# it.	48	6	5	27	26	4	19	8	7	4	17	3	7	6	6	3
	t (s)	25.41	26.08	26.20	23.57	21.32	21.92	18.41	18.01	16.64	16.86	14.92	15.41	15.58	15.45	14.35	14.37
10 %	# it.	35	5	3	27	14	3	9	4	4	4	15	3	4	4	5	3
	t (s)	27.31	24.44	27.12	23.57	22.82	26.05	18.81	18.77	17.42	16.86	16.31	15.41	16.36	16.10	15.44	14.37
Init	# it.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	t (s)	51.92	51.92	51.92	51.92	36.98	36.98	36.98	36.98	36.29	36.29	36.29	36.29	34.84	34.84	34.84	34.84
% improv.		−51.85	-51.88	−51.86	-55.03	−43.45	-43.48	−52.42	-53.12	-55.61	−55.59	-58.93	−58.86	−56.94	-57.28	−59.32	-59.74

The initial and final values, together with the intermediate values at 1%, 5% and 10% of the final value are provided along with their required number of iterations
Minimum values highlighted in bold

Table 13 Total time of the trajectory tracking computed by the trajectory planner for the second evaluation path

Velocity		S				M-S				M-F				F			
Distance		Acc		Ina		Acc		Ina		Acc		Ina		Acc		Ina	
Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
Fin	# it.	75	87	49	55	27	84	54	23	45	46	18	20	23	46	28	16
	t (s)	25.52	25.59	24.68	24.29	19.07	18.93	18.40	18.44	17.45	18.54	16.75	16.82	16.72	16.57	15.81	15.87
1 %	# it.	40	39	21	21	13	47	23	8	11	8	9	10	4	9	9	8
	t (s)	25.76	25.75	24.89	24.29	19.25	19.07	18.52	18.61	17.62	18.67	16.87	16.96	16.80	16.72	15.96	15.97
5 %	# it.	6	33	7	6	7	16	6	8	8	7	4	8	3	3	4	3
	t (s)	25.46	26.47	25.54	25.36	19.34	19.79	19.27	18.61	18.21	18.94	17.36	17.03	17.07	17.33	16.47	16.53
10 %	# it.	4	15	3	4	5	5	3	5	4	3	3	3	3	3	3	3
	t (s)	27.75	27.74	26.60	25.94	20.22	20.60	20.20	20.03	19.00	20.14	17.77	18.34	17.07	17.33	17.38	16.53
Init	# it.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	t (s)	58.28	58.28	58.28	58.28	43.36	43.36	43.36	43.36	42.31	42.31	42.31	42.31	40.63	40.63	40.63	40.63
% improv.		−56.21	−56.09	−57.65	−58.32	−56.02	−56.34	−57.56	−57.47	−58.76	−56.18	−60.41	−60.25	−58.85	−59.22	−61.09	−60.94

The initial and final values, together with the intermediate values at 1%, 5% and 10% of the final value are provided along with their required number of iterations
Minimum values highlighted in bold

Table 14 Percentage of the configuration parameters with respect to their maximum value for the magnitudes of the initial trajectories of the first evaluation path

Velocity		**S**	**M-S**	**M-F**	**F**
Max. d		0.0	0.0	0.0	0.0
Vel.	**Lin.**	54.02	55.79	49.81	45.90
	Ang.	43.61	45.04	40.21	37.05
Acc.	**Lin.**	39.17	42.53	38.50	35.61
	Ang.	31.62	34.34	31.08	28.75
Jer.	**Lin.**	31.10	35.50	31.10	30.24
	Ang.	25.11	28.66	25.10	24.42
Sna.	**Lin.**	100.0	100.0	83.75	78.04
	Ang.	80.73	80.73	67.61	63.00
Cra.	**Lin.**	82.21	100.0	100.0	100.0
	Ang.	62.45	78.78	80.73	80.73
Pop	**Lin.**	20.27	28.81	28.88	30.30
	Ang.	14.90	21.35	21.94	23.10
Average	**Lin.**	54.46	60.44	55.34	53.35
	Ang.	43.00	48.15	44.45	42.83
	Total	48.73	54.29	49.89	48.09
$\mathbf{u_x}$	**Min.**	28.04	43.24	44.35	46.82
	Max.	28.04	43.24	44.35	46.82
$\mathbf{u_y}$	**Min.**	23.11	38.13	40.14	42.58
	Max.	14.81	21.84	22.19	23.31
$\mathbf{u_z}$	**Min.**	5.73	8.82	9.03	9.52
	Max.	4.74	7.52	7.88	8.31
$\mathbf{u_\psi}$	**Min.**	34.55	54.78	57.43	60.59
	Max.	9.56	16.06	17.11	18.13
Average		18.57	29.20	30.31	32.01
Average		34.92	42.15	40.06	39.67

Minimum values highlighted in bold

during the optimization process for the first and second evaluation paths respectively.

Tables 12 and 13 show the summarized information of the total time of the trajectory tracking computed during the optimization process of the trajectory planner for the first and second evaluation paths respectively.

Tables 14 and 15 gather the percentage of the configuration parameters with respect to their maximum value for the magnitudes of the initial trajectories of the first and second evaluation paths respectively.

Tables 16 and 17 collect the percentage of the configuration parameters with respect to their maximum value for the magnitudes of the planned trajectories of the first and second evaluation paths respectively.

Table 15 Percentage of the configuration parameters with respect to their maximum value for the magnitudes of the initial trajectories of the second evaluation path

Velocity		S	M-S	M-F	F
Max. d		0.0	0.0	0.0	0.0
Vel.	**Lin.**	56.84	58.71	53.31	49.12
	Ang.	41.45	42.81	38.87	35.82
Acc.	**Lin.**	40.53	44.00	40.51	37.47
	Ang.	29.55	32.08	29.54	27.32
Jer.	**Lin.**	31.63	36.10	32.17	31.29
	Ang.	23.07	26.33	23.46	22.82
Sna.	**Lin.**	100.0	100.0	85.18	79.38
	Ang.	72.92	72.92	62.11	57.88
Cra.	**Lin.**	100.0	100.0	100.0	100.0
	Ang.	55.45	69.95	72.92	72.92
Pop	**Lin.**	30.53	35.66	35.75	37.51
	Ang.	12.92	18.64	19.48	20.44
Average	**Lin.**	59.92	62.41	57.82	55.79
	Ang.	39.23	43.79	41.06	39.53
	Total	49.57	53.10	49.44	47.66
$\mathbf{u_x}$	**Min.**	25.27	41.67	44.70	47.42
	Max.	29.23	45.86	47.03	49.64
$\mathbf{u_y}$	**Min.**	18.66	28.91	30.62	32.25
	Max.	21.73	34.14	36.31	38.23
$\mathbf{u_z}$	**Min.**	12.16	15.36	15.74	16.61
	Max.	4.48	7.09	7.56	7.97
$\mathbf{u_\psi}$	**Min.**	32.59	51.65	55.07	58.09
	Max.	20.07	31.25	31.98	33.74
Average		20.52	31.99	33.63	35.49
Average		36.15	42.53	41.06	40.76

Minimum values highlighted in bold

Table 16 Percentage of the configuration parameters with respect to their maximum value for the magnitudes of the planned trajectories of the first evaluation path

Velocity		S				M-S				M-F				F			
Distance		Acc		Ina		Acc		Ina		Acc		Ina		Acc		Ina	
Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
Max. d		99.89	99.97	57.59	64.30	99.95	99.74	59.05	67.65	99.99	100.0	83.37	71.48	100.0	100.0	65.39	75.56
Vel.	**Lin.**	99.99	100.0	99.99	99.98	100.0	99.99	99.99	100.0	100.0	100.0	99.99	99.99	99.99	99.99	99.99	99.99
	Ang.	77.59	74.76	76.26	76.27	74.15	74.95	73.51	73.97	83.83	83.43	77.14	73.95	75.04	78.58	73.36	78.16
Acc.	**Lin.**	98.38	100.0	97.20	95.80	100.0	99.98	100.0	99.99	100.0	100.0	99.99	99.98	99.89	99.99	97.01	99.31
	Ang.	49.79	33.01	33.63	34.86	33.86	34.51	35.82	33.10	48.08	45.18	35.53	35.95	46.33	44.70	41.53	43.70
Jer.	**Lin.**	53.73	55.59	59.87	61.50	68.89	64.27	57.39	62.29	75.16	74.71	67.79	60.89	86.64	90.90	86.58	63.95
	Ang.	74.88	29.66	34.90	26.89	46.16	61.03	51.76	47.75	70.12	73.52	45.12	39.90	60.91	60.47	60.08	51.44
Sna.	**Lin.**	99.98	99.98	99.99	99.94	99.98	99.84	99.99	100.0	99.97	100.0	99.97	99.98	99.97	100.0	99.98	99.98
	Ang.	91.33	63.62	72.54	72.19	78.50	85.39	96.14	76.77	85.53	98.36	71.72	65.00	98.98	96.29	79.69	82.39
Cra.	**Lin.**	75.82	74.46	72.85	72.53	100.0	100.0	99.70	95.41	99.89	100.0	99.78	98.67	99.95	100.0	99.99	99.30
	Ang.	76.92	65.77	57.40	51.55	75.23	78.00	65.30	76.08	92.29	91.46	90.92	75.00	98.33	99.95	90.85	95.83
Pop	**Lin.**	20.70	20.63	20.60	17.10	30.11	29.69	37.85	34.05	31.14	29.23	43.38	30.39	30.48	29.66	36.15	52.76
	Ang.	24.57	28.81	14.87	14.89	19.09	21.12	16.67	21.99	25.47	20.32	26.02	24.50	30.98	34.31	22.05	24.75
Average	**Lin.**	74.77	75.44	75.08	74.48	83.16	82.30	82.49	81.96	84.36	83.99	85.15	81.65	85.15	86.74	86.62	85.88
	Ang.	65.85	48.11	48.27	46.11	54.50	59.17	56.53	54.94	67.55	68.71	57.74	52.38	68.43	69.05	61.26	62.71
	Total	70.31	61.77	61.67	60.29	68.83	70.73	69.51	68.45	75.96	76.35	71.45	67.02	76.79	77.90	73.94	74.30
$\mathbf{u_x}$	**Min.**	51.84	61.24	58.95	52.62	93.30	93.59	89.13	91.81	99.91	100.0	99.93	99.97	100.0	100.0	99.99	100.0
	Max.	52.65	53.26	54.48	54.39	83.37	83.38	84.13	89.32	96.04	98.14	99.68	99.99	99.98	100.0	100.0	99.98
$\mathbf{u_y}$	**Min.**	50.42	50.61	51.30	51.52	78.06	78.01	84.25	80.60	95.43	96.94	98.72	99.78	99.99	99.96	99.98	99.82
	Max.	30.63	30.27	30.86	34.51	41.79	41.35	58.86	53.86	47.56	46.31	70.12	69.16	63.79	61.62	82.98	67.19
$\mathbf{u_z}$	**Min.**	11.21	10.95	6.75	10.66	16.78	16.85	12.11	13.81	20.11	19.94	19.97	19.85	22.07	22.69	20.81	22.38
	Max.	7.22	7.40	9.64	12.59	12.22	12.06	13.68	10.50	14.29	14.35	13.41	13.66	15.32	15.13	17.04	17.73
$\mathbf{u_\psi}$	**Min.**	54.01	50.78	47.31	46.73	79.24	80.88	72.10	76.72	99.94	99.93	93.97	81.48	99.95	99.99	99.97	97.17
	Max.	10.96	8.06	7.28	8.43	13.68	13.98	11.68	13.17	17.14	21.81	14.53	15.26	16.45	14.25	17.28	17.29
Average		33.62	30.07	33.32	33.93	52.31	52.51	53.24	53.72	61.30	62.18	63.79	62.39	64.69	64.21	67.26	65.20
Average		57.74	53.04	50.68	50.44	64.02	65.17	62.81	62.80	71.52	72.08	69.10	65.47	73.29	73.73	70.99	70.89

Minimum values highlighted in bold

Table 17 Percentage of the configuration parameters with respect to their maximum value for the magnitudes of the planned trajectories of the second evaluation path

Velocity		S				M-S				M-F				F			
Distance		Acc		Ina		Acc		Ina		Acc		Ina		Acc		Ina	
Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
Max. d		99.98	99.99	72.08	66.02	99.98	99.99	79.41	48.25	100.0	99.99	100.0	98.33	100.0	100.0	50.07	48.39
Vel.	**Lin.**	100.0	100.0	100.0	100.0	99.97	100.0	99.99	100.0	100.0	100.0	100.0	99.99	99.99	99.99	99.96	100.0
	Ang.	70.89	70.52	72.17	69.80	69.31	72.12	70.18	72.12	68.23	69.51	68.84	68.50	72.24	65.80	68.89	73.72
Acc.	**Lin.**	100.0	99.97	99.98	99.98	100.0	100.0	99.99	99.96	99.99	100.0	99.99	99.99	99.99	99.68	99.94	99.99
	Ang.	69.09	66.62	67.94	67.70	66.84	68.48	70.23	66.15	64.79	63.57	67.27	64.26	63.08	57.54	55.91	66.04
Jer.	**Lin.**	54.83	54.57	70.00	57.13	65.90	58.46	72.35	85.96	68.51	98.13	83.08	86.42	73.45	66.21	72.92	92.16
	Ang.	41.65	41.81	43.19	41.45	54.34	54.35	44.03	39.87	52.88	53.82	38.43	38.24	73.10	44.82	41.15	54.14
Sna.	**Lin.**	99.99	99.99	99.96	99.96	99.98	100.0	99.98	99.99	100.0	99.97	99.99	100.0	100.0	99.99	99.98	99.98
	Ang.	99.86	97.25	98.68	99.09	92.57	81.94	89.93	99.98	86.22	86.13	89.20	77.21	99.95	90.90	99.53	99.98
Cra.	**Lin.**	86.49	92.07	94.67	72.95	99.96	99.95	99.99	99.09	100.0	99.98	99.99	99.99	99.99	99.99	99.96	99.98
	Ang.	91.48	57.03	68.99	99.26	91.69	97.74	79.89	99.74	99.94	99.96	97.91	88.53	99.83	99.81	99.36	99.97
Pop	**Lin.**	28.42	25.64	41.00	23.42	33.91	33.87	47.68	34.10	35.36	33.82	38.55	42.80	39.73	35.00	39.57	35.46
	Ang.	28.40	18.27	18.13	31.22	20.10	19.91	22.60	45.68	30.03	32.87	40.05	24.13	30.30	37.56	44.64	43.80
Average	**Lin.**	78.29	78.71	84.27	75.57	83.29	82.05	86.66	86.52	83.98	88.65	86.93	88.20	85.51	83.48	85.39	87.93
	Ang.	66.90	58.58	61.52	68.09	65.81	67.76	62.81	70.59	67.02	67.64	66.95	60.15	73.08	65.92	66.25	72.94
	Total	72.59	68.65	72.89	71.83	74.55	73.90	74.74	78.55	75.50	78.15	76.94	74.17	79.30	74.70	76.82	80.44
$\mathbf{u_x}$	**Min.**	53.44	50.83	50.20	54.28	80.20	77.00	79.21	83.81	94.43	96.93	95.08	98.55	99.99	99.87	99.92	99.97
	Max.	63.34	63.29	61.64	62.04	96.55	96.66	96.04	81.95	100.0	100.0	100.0	99.93	99.78	99.99	99.98	99.97
$\mathbf{u_y}$	**Min.**	32.91	32.89	44.77	34.68	49.35	49.37	53.30	49.49	56.56	57.17	81.76	76.65	64.45	61.08	63.31	64.86
	Max.	50.27	50.13	46.77	53.16	80.35	83.00	83.22	76.69	94.53	75.86	89.16	87.93	98.48	96.87	90.96	87.55
$\mathbf{u_z}$	**Min.**	26.67	26.27	30.39	28.33	32.82	33.64	32.95	19.94	36.15	35.78	29.08	41.66	36.08	35.96	40.10	43.29
	Max.	6.87	6.84	9.58	8.03	10.37	10.76	10.36	14.78	12.55	11.96	12.55	13.73	13.90	14.18	13.61	25.93
$\mathbf{u_\psi}$	**Min.**	53.86	57.56	59.05	48.78	80.98	88.50	86.18	86.63	98.70	97.23	98.20	97.33	99.20	99.80	99.68	98.58
	Max.	40.97	40.69	42.69	42.37	59.06	55.45	61.26	61.78	82.78	91.31	86.78	78.52	99.68	72.31	77.03	75.00
Average		41.04	41.06	43.14	41.46	61.21	61.80	62.82	61.88	71.96	70.78	74.08	74.29	76.45	72.51	73.07	74.39
Average		61.88	59.63	61.52	59.98	70.68	70.53	70.42	70.76	75.32	76.38	76.95	75.37	79.20	75.07	74.12	76.61

Minimum values highlighted in bold

Table 18 Percentage of the energy (88) of the planned trajectory magnitudes with respect to the initial trajectories of the first evaluation path

Velocity		S				M-S				M-F				F			
Distance		Acc		Ina		Acc		Ina		Acc		Ina		Acc		Ina	
Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
Vel.	**Lin.**	95.56	108.59	21.39	94.02	80.83	70.07	13.31	18.33	47.79	40.74	52.94	45.43	109.55	124.24	131.93	164.41
	Ang.	90.82	88.67	91.54	99.04	70.50	71.48	84.73	86.55	101.66	102.85	112.97	112.33	112.00	115.45	117.05	120.22
Acc.	**Lin.**	31.98	40.43	−11.93	30.91	27.05	23.47	−16.62	−10.21	9.13	4.47	9.15	6.39	42.00	47.30	50.38	75.53
	Ang.	27.11	−25.63	−39.10	−39.66	−3.01	14.29	−34.42	−35.44	52.13	70.84	−7.75	−1.48	75.17	90.29	40.77	45.00
Jer.	**Lin.**	−9.36	−4.90	−33.49	−9.98	−9.42	−11.28	−37.31	−30.77	−20.17	−23.25	−22.61	−22.79	−3.58	−3.30	−3.20	14.67
	Ang.	44.96	−34.10	−40.39	−51.61	3.07	33.13	−22.79	−27.86	77.47	103.85	1.32	13.27	125.84	161.02	63.49	61.97
Sna.	**Lin.**	−27.65	−26.13	−41.13	−27.86	−25.17	−25.90	−44.56	−37.70	−31.93	−34.21	−36.23	−35.11	−21.97	−23.71	−26.12	−13.78
	Ang.	16.74	−43.41	−47.75	−55.90	−15.48	14.98	−25.66	−27.86	48.48	72.59	−13.27	1.36	108.21	147.64	38.92	39.79
Cra.	**Lin.**	−29.62	−29.97	−36.70	−29.37	−26.49	−26.50	−40.18	−33.32	−30.60	−32.21	−36.40	−34.39	−24.25	−26.19	−30.15	−23.23
	Ang.	−31.63	−49.98	−57.66	−60.43	−40.92	−34.49	−52.87	−49.84	−17.51	−21.85	−40.75	−32.45	−4.64	13.11	−33.18	−24.31
Pop	**Lin.**	−18.29	−18.95	−19.70	−17.00	−16.43	−16.23	−22.71	−18.58	−18.79	−19.50	−22.11	−20.16	−16.68	−17.58	−19.39	−17.40
	Ang.	−28.16	−38.42	−48.82	−51.17	−37.99	−40.87	−57.01	−46.44	−28.70	−42.73	−37.60	−30.58	−24.61	−9.76	−49.47	−42.78
Average.	**Lin.**	7.10	11.51	−20.26	6.79	5.06	3.10	−24.68	−18.58	−7.43	−10.66	−9.21	−10.10	14.18	16.79	17.24	33.37
	Ang.	19.97	−17.15	−23.90	−26.62	−3.97	9.75	−18.00	−16.47	38.92	47.59	2.49	10.41	65.33	86.29	29.60	33.32
	Total	13.54	−2.82	−21.98	−9.92	0.54	6.43	−21.34	−17.53	15.75	18.47	−3.36	0.15	39.75	51.54	23.42	33.34

Minimum values highlighted in bold

Table 19 Percentage of the energy (88) of the planned trajectory magnitudes with respect to the initial trajectories of the second evaluation path.1

Velocity		S				M-S				M-F				F			
Distance		Acc		Ina		Acc		Ina		Acc		Ina		Acc		Ina	
Rot. error		$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
Vel.	**Lin.**	20.55	20.76	41.81	33.10	−56.58	−56.90	−53.43	−36.49	−63.76	−63.46	−58.96	−56.18	−47.86	−61.74	−53.11	−51.46
	Ang.	99.44	98.38	103.79	105.84	88.80	90.09	94.43	91.50	107.07	99.04	106.90	103.53	119.61	110.32	110.82	116.58
Acc.	**Lin.**	−8.51	−8.37	11.73	4.08	−58.29	−59.03	−51.63	−40.29	−61.38	−61.64	−57.52	−53.50	−45.30	−57.47	−49.80	−49.06
	Ang.	301.02	269.69	276.25	254.63	238.23	236.64	237.77	245.18	311.18	326.98	311.81	277.25	378.35	295.10	287.71	355.58
Jer.	**Lin.**	−29.24	−29.03	−12.19	−18.02	−58.96	−60.16	−50.50	−44.42	−59.18	−60.03	−56.76	−52.28	−45.40	−54.46	−49.16	−48.70
	Ang.	161.73	129.07	140.28	105.14	115.83	109.98	110.28	118.82	163.40	196.05	106.69	129.43	249.53	163.07	144.39	243.85
Sna.	**Lin.**	−37.63	−37.37	−24.86	−28.50	−54.53	−56.10	−46.17	−43.95	−52.95	−53.97	−52.31	−47.89	−42.28	−48.51	−46.11	−45.45
	Ang.	74.13	42.33	73.17	48.44	53.02	44.01	43.81	58.83	88.08	99.98	85.13	59.54	219.53	98.61	112.65	223.44
Cra.	**Lin.**	−35.17	−35.31	−28.78	−29.55	−43.88	−45.37	−38.61	−38.77	−42.30	−41.91	−43.32	−40.32	−36.09	−39.11	−40.23	−39.09
	Ang.	−4.90	−28.08	−11.86	4.75	−8.81	−12.87	−9.91	14.53	−6.15	1.75	29.78	2.53	91.03	17.94	60.11	81.02
Pop.	**Lin.**	−20.33	−21.98	−21.61	−18.59	−24.54	−25.19	−23.46	−24.48	−25.89	−21.60	−25.47	−25.09	−23.24	−24.73	−26.09	−25.28
	Ang.	−23.77	−52.59	−42.51	13.48	−49.96	−42.79	−28.25	1.73	−21.35	−15.98	2.27	−25.78	−10.36	−8.53	−14.27	−2.93
Average	**Lin.**	−18.39	−18.55	−5.65	−9.58	−49.46	−50.46	−43.97	−38.07	−50.91	−50.44	−49.06	−45.88	−40.03	−47.67	−44.08	−43.17
	Ang.	101.27	76.47	89.85	88.71	72.85	70.84	74.69	88.43	107.04	117.97	116.26	91.08	−10.36	112.75	116.90	169.59
	Total	41.44	28.96	42.10	39.57	11.69	10.19	15.36	25.18	28.06	33.77	33.60	22.60	67.29	32.54	36.41	63.21

Minimum values highlighted in bold

Table 20 Percentage of the energy (88) of the planned control commands references with respect to the initial trajectories of the first evaluation path

Velocity	S				M-S				M-F				F			
Distance	Acc		Ina		Acc		Ina		Acc		Ina		Acc		Ina	
Rot. error	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
$\mathbf{u_x}$	175.62	174.73	165.22	179.56	166.21	164.01	162.92	164.52	249.86	255.51	230.76	231.95	275.94	260.18	252.08	268.09
$\mathbf{u_y}$	107.11	111.35	116.73	132.96	109.18	108.19	157.03	163.87	180.56	178.77	243.17	230.55	229.41	276.93	277.13	241.83
$\mathbf{u_z}$	98.08	101.83	73.18	135.52	99.85	99.90	84.83	87.50	136.25	134.46	134.00	132.46	147.92	151.39	146.34	149.35
$\mathbf{u_\psi}$	85.39	78.66	80.08	86.99	60.94	64.19	69.06	70.53	94.82	98.87	96.30	96.45	107.85	112.89	106.78	110.49
Average	116.55	116.64	108.80	133.76	109.05	109.07	118.46	121.60	165.37	166.90	176.06	172.85	190.28	200.35	195.58	192.44

Minimum values highlighted in bold

Table 21 Percentage of the energy (88) of the planned control commands references with respect to the initial trajectories of the second evaluation path

Velocity	S				M-S				M-F				F			
Distance	Acc		Ina		Acc		Ina		Acc		Ina		Acc		Ina	
Rot. error	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$	$\delta\psi$	$\delta\theta$
$\mathbf{u_x}$	147.79	148.37	152.67	158.62	156.47	153.91	162.88	153.47	199.24	200.04	199.60	200.57	203.14	213.85	228.97	217.44
$\mathbf{u_y}$	153.62	151.01	176.79	178.36	154.70	162.46	184.98	154.70	191.07	156.12	241.57	228.88	202.21	216.13	192.50	205.35
$\mathbf{u_z}$	154.18	151.52	248.43	225.39	146.94	150.90	200.95	281.30	161.77	159.86	155.77	291.54	161.48	153.24	241.26	339.86
$\mathbf{u_\psi}$	118.62	114.68	120.21	120.15	110.28	110.84	114.57	113.30	136.80	131.99	136.82	128.96	159.08	139.20	138.35	153.77
Average	143.55	141.39	174.53	170.63	142.10	144.53	165.84	175.69	172.22	162.00	183.44	212.49	181.48	180.61	200.27	229.11

Minimum values highlighted in bold

Tables 18 and 19 display the percentage of the energy (computed with Eq. 88) of the planned trajectory magnitudes with respect to the initial trajectories of the first and second evaluation paths respectively.

Tables 20 and 21 show the percentage of the energy (88) of the planned control commands references with respect to the initial trajectories of the first and second evaluation paths respectively.

References

1. Achtelik, M.W., Lynen, S., Chli, M., Siegwart, R.: Inversion Based Direct Position Control and Trajectory Following for Micro Aerial Vehicles. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2933–2939 (2013). https://doi.org/10.1109/IROS.2013.CR1
2. Achtelik, M.W., Lynen, S., Chli, M., Siegwart, R.: Inversion Based Direct Position Control and Trajectory Following for Micro Aerial Vehicles. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2933–2939. IEEE (2013)
3. Aguiar, A.P., Hespanha, J.P.: Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. IEEE Trans. Autom. Control **52**(8), 1362–1379 (2007). https://doi.org/10.1109/TAC.2007.902731
4. Alvarenga, J., Vitzilaios, N.I., Valavanis, K.P., Rutherford, M.J.: Survey of unmanned helicopter model-based navigation and control techniques. J. Intell. Robot. Syst. **80**(1), 87–138 (2015). https://doi.org/10.1007/s10846-014-0143-5
5. Beul, M., Behnke, S.: Analytical Time-Optimal Trajectory Generation and Control for Multirotors. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 87–96 (2016). https://doi.org/10.1109/ICUAS.2016.7502532
6. Beul, M., Behnke, S.: Fast Full State Trajectory Generation for Multirotors. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 408–416 (2017). 10.1109/ICUAS.2017.7991304
7. Boeuf, A., Cortés, J., Alami, R., Siméon, T.: Planning Agile Motions for Quadrotors in Constrained Environments. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 218–223 (2014). https://doi.org/10.1109/IROS.2014.6942564
8. Brescianini, D., D'Andrea, R.: Computationally efficient trajectory generation for fully actuated multirotor vehicles. IEEE Trans. Robot. **34**(3), 555–571 (2018). https://doi.org/10.1109/TRO.2018.2813373
9. Carrio, A., Pestana, J., Sanchez-Lopez, J.L., Suarez-Fernandez, R., Campoy, P., Tendero, R., García-De-Viedma, M., González-Rodrigo, B., Bonatti, J., Rejas-Ayuga, J.G., Martínez-Marín, R., Marchamalo-Sacristán, M.: UBRISTES: UAV-Based Building Rehabilitation with Visible and Thermal Infrared Remote Sensing, pp. 245–256. Springer International Publishing, Cham (2016)
10. Castillo-Lopez, M., Olivares-Mendez, M.A., Voos, H.: Evasive Maneuvering for Uavs: an Mpc Approach. In: Ollero, A., Sanfeliu, A., Montano, L., Lau, N., Cardeira, C. (eds.) ROBOT 2017: Third Iberian Robotics Conference, pp. 829–840. Springer International Publishing, Cham (2018)
11. Chen, H., Wang, X., Li, Y.: A Survey of Autonomous Control for Uav. In: 2009 International Conference on Artificial Intelligence and Computational Intelligence, vol. 2, pp. 267–271 (2009). https://doi.org/10.1109/AICI.2009.147
12. Consolini, L., Locatelli, M., Minari, A., Piazzi, A.: An optimal complexity algorithm for minimum-time velocity planning, vol. 103, pp. 50–57 (2017). https://doi.org/10.1016/j.sysconle.2017.02.001. http://www.sciencedirect.com/science/article/pii/S0167691117300245
13. Diebel, J.: Representing attitude: Euler angles, unit quaternions and rotation vectors (2006)
14. Diehl, M., Ferreau, H.J., Haverbeke, N.: Efficient Numerical Methods for Nonlinear Mpc and Moving Horizon Estimation. In: Nonlinear Model Predictive Control, pp. 391–417. Springer (2009)
15. collab=Dvořák, J., de Lellis, M., Hurák, Z.: Advanced Control of Quadrotor Using Eigenaxis Rotation. In: 2011 IEEE International Conference on Control Applications (CCA), pp. 153–158. IEEE (2011)
16. Ezair, B., Tassa, T., Shiller, Z.: Planning high order trajectories with general initial and final conditions and asymmetric bounds. Int. J. Robot. Res. **33**(6), 898–916 (2014). https://doi.org/10.1177/0278364913517148
17. Faessler, M., Franchi, A., Scaramuzza, D.: Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. IEEE Robot. Autom. Letters **3**(2), 620–626 (2018). https://doi.org/10.1109/LRA.2017.2776353
18. Falanga, D., Foehn, P., Lu, P., Scaramuzza, D.: Pampc: Perception-aware Model Predictive Control for Quadrotors. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–8. IEEE (2018)
19. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous uav guidance. J. Intell. Robot. Syst. **57**(1), 65 (2009). 10.1007/s10846-009-9383-1
20. Hamilton, W.R. In: Hamilton, W.E. (ed.): Elements of Quaternions https://doi.org/10.1017/CBO9780511707162, Cambridge Library Collection - Mathematics. Cambridge University Press, Cambridge (2010)
21. Haschke, R., Weitnauer, E., Ritter, H.: On-Line Planning of Time-Optimal, Jerk-Limited Trajectories. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3248–3253 (2008). 10.1109/IROS.2008.4650924
22. Hauser, K., Ng-Thow-Hing, V.: Fast Smoothing of Manipulator Trajectories Using Optimal Bounded-Acceleration Shortcuts. In: 2010 IEEE International Conference on Robotics and Automation, pp. 2493–2498 (2010). https://doi.org/10.1109/ROBOT.2010.5509683
23. Hönig, W., Preiss, J.A., Kumar, T.K.S., Sukhatme, G.S., Ayanian, N.: Trajectory planning for quadrotor swarms. IEEE Trans. Robot. **34**(4), 856–869 (2018). https://doi.org/10.1109/TRO.2018.2853613
24. Horn, B.K.: Closed-form solution of absolute orientation using unit quaternions. Josa a **4**(4), 629–642 (1987)
25. Houska, B. et al.: ACADO Toolkit – an open source framework for automatic control and dynamic optimization optimal control applications and methods (2011)
26. Judd, K.B., McLain, T.W.: Spline Based Path Planning for Unmanned Air Vehicles. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, vol. 9. Montreal, Canada (2001)
27. Jung, D., Tsiotras, P.: On-Line Path Generation for Small Unmanned Aerial Vehicles Using B-Spline Path Templates. In: AIAA Guidance, Navigation and Control Conference, IEEE, vol. 7135 (2008)
28. Kamel, M., Alexis, K., Achtelik, M., Siegwart, R.: Fast Nonlinear Model Predictive Control for Multicopter Attitude Tracking on So (3). In: 2015 IEEE Conference on Control Applications (CCA), pp. 1160–1166. IEEE (2015)

29. Kamel, M., Burri, M., Siegwart, R.: Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles. IFAC-PapersOnLine **50**(1), 3463–3469 (2017)
30. Kerrigan, E.C., Maciejowski, J.M.: Soft Constraints and Exact Penalty Functions in Model Predictive Control. In: Proc. UKACC International Conference (Control. Citeseer (2000)
31. Kuipers, J.B., et al.: Quaternions and rotation sequences, vol. 66. Princeton University Press, Princeton (1999)
32. Lee, H., Kim, H.J.: Trajectory tracking control of multirotors from modelling to experiments: A survey. Int. J. Control Autom. Syst. **15**(1), 281–292 (2017)
33. Lee, T.: Exponential stability of an attitude tracking control system on so(3) for large-angle rotational maneuvers. Syst. Control Letters **61**, 231–237 (2012). https://doi.org/10.1016/j.sysconle.2011.10.017. http://www.sciencedirect.com/science/article/pii/S0167691111002829
34. Lee, T., Leoky, M., McClamroch, N.H.: Geometric Tracking Control of a Quadrotor Uav on Se (3). In: Decision and Control (CDC), 2010 49Th IEEE Conference On, pp. 5420–5425. IEEE (2010)
35. Li, Y., Song, S.: A Survey of Control Algorithms for Quadrotor Unmanned Helicopter. In: 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), pp. 365–369 (2012). https://doi.org/10.1109/ICACI.2012.6463187
36. Macfarlane, S., Croft, E.A.: Jerk-bounded manipulator trajectory planning: design for real-time applications. IEEE Trans. Robot. Autom. **19**(1), 42–52 (2003). https://doi.org/10.1109/TRA.2002.807548
37. Manyam, S.G., Rathinam, S., Casbeer, D., Garcia, E.: Tightly bounding the shortest dubins paths through a sequence of points. J. Intell. Robot. Syst. **88**(2), 495–511 (2017). https://doi.org/10.1007/s10846-016-0459-4
38. Mellinger, D., Kumar, V.: Minimum Snap Trajectory Generation and Control for Quadrotors. In: 2011 IEEE International Conference on Robotics and Automation, pp. 2520–2525 (2011). https://doi.org/10.1109/ICRA.2011.5980409
39. Mellinger, D., Kumar, V.: Minimum Snap Trajectory Generation and Control for Quadrotors. In: Robotics and Automation (ICRA), 2011 IEEE International Conference On, pp. 2520–2525. IEEE (2011)
40. Mohammed, F., Idries, A., Mohamed, N., Al-Jaroodi, J., Jawhar, I.: Uavs for Smart Cities: Opportunities and Challenges. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference On, pp. 267–273 (2014). https://doi.org/10.1109/ICUAS.2014.6842265
41. Morari, M., Lee, J.H.: Model predictive control: past, present and future. Comput. Chem. Eng. **23**(4-5), 667–682 (1999)
42. Mueller, M.W., D'Andrea, R.: A Model Predictive Controller for Quadrocopter State Interception. In: 2013 European Control Conference (ECC), pp. 1383–1389 (2013)
43. Mueller, M.W., Hehn, M., D'Andrea, R.: A Computationally Efficient Algorithm for State-To-State Quadrocopter Trajectory Generation and Feasibility Verification. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3480–3486 (2013). https://doi.org/10.1109/IROS.2013.6696852
44. Neunert, M., De Crousaz, C., Furrer, F., Kamel, M., Farshidian, F., Siegwart, R., Buchli, J.: Fast Nonlinear Model Predictive Control for Unified Trajectory Optimization and Tracking. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1398–1404. IEEE (2016)
45. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an Open-Source Robot Operating System. In: ICRA Workshop on Open Source Software, vol. 3, p. 5 (2009)
46. Richter, C., Bry, A., Roy, N.: Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments, pp. 649–666. Springer International Publishing, Cham (2016)
47. Sanchez-Lopez, J.L., Arellano-Quintana, V., Tognon, M., Campoy, P., Franchi, A.: Visual marker based multi-sensor fusion state estimation. IFAC-PapersOnLine **50**(1), 16,003–16,008 (2017). https://doi.org/10.1016/j.ifacol.2017.08.1911. 20th IFAC World Congress
48. Sanchez-Lopez, J.L., Fernández, R.A.S., Bavle, H., Sampedro, C., Molina, M., Pestana, J., Campoy, P.: Aerostack: an Architecture and Open-Source Software Framework for Aerial Robotics. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 332–341 (2016). https://doi.org/10.1109/ICUAS.2016.7502591
49. Sanchez-Lopez, J.L., Molina, M., Bavle, H., Sampedro, C., Suárez Fernández, R.A., Campoy, P.: A multi-layered component-based approach for the development of aerial robotic systems: the aerostack framework. J. Intell. Robot. Syst., 1–27 (2017)
50. Sanchez-Lopez, J.L., Olivares-Mendez, M.A., Castillo-Lopez, M., Voos, H.: Towards Trajectory Planning from a Given Path for Multirotor Aerial Robots Trajectory Tracking. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1342–1351 (2018). https://doi.org/10.1109/ICUAS.2018.8453428
51. Sanchez-Lopez, J.L., Pestana, J., Campoy, P.: A Robust Real-Time Path Planner for the Collision-Free Navigation of Multirotor Aerial Robots in Dynamic Environments. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 316–325 (2017). https://doi.org/10.1109/ICUAS.2017.7991354
52. Sanchez-Lopez, J.L., Wang, M., Olivares-Mendez, M.A., Molina, M., Voos, H.: A real-time 3d path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments. J. Intell. Robot. Syst. **93**(1), 33–53 (2019). https://doi.org/10.1007/s10846-018-0809-5
53. Santos, M.C.P., Rosales, C.D., Sarapura, J.A., Sarcinelli-Filho, M., Carelli, R.: An adaptive dynamic controller for quadrotor to perform trajectory tracking tasks. J. Intell. Robot. Syst. **93**(1), 5–16 (2019). 10.1007/s10846-018-0799-3
54. Sola, J.: Quaternion kinematics for the error-state kalman filter. arXiv:1711.02508 (2017)
55. Tang, S., Thomas, J., Kumar, V.: Hold or take optimal plan (hoop): a quadratic programming approach to multi-robot trajectory generation. Int. J. Robot. Res. **37**(9), 1062–1084 (2018). https://doi.org/10.1177/0278364917741532
56. Tayebi, A., McGilvray, S.: Attitude Stabilization of a Four-Rotor Aerial Robot. In: 2004 43Rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601), vol. 2, pp. 1216–1221. IEEE (2004)
57. Valavanis, K.P., Vachtsevanos, G.J.: Handbook of unmanned aerial vehicles. Springer, Berlin (2015)
58. Yang, K., Sukkarieh, S.: An analytical continuous-curvature path-smoothing algorithm. IEEE Trans. Robot. **26**(3), 561–568 (2010). https://doi.org/10.1109/TRO.2010.2042990
59. Zhang, F.: Quaternions and matrices of quaternions. Linear Algebra Its Appl. **251**, 21–57 (1997)

Dr. Jose Luis Sanchez-Lopez is a post-doctoral research associate at the Automation & Robotics Research Group of the Interdisciplinary Centre for Security Reliability and Trust (SnT) of the University of Luxembourg since June 2017. He received his Ph. D. in Robotics in 2017 at the Technical University of Madrid, after been a visiting researcher at the Arizona State University (AZ, USA), and the at LAAS-CNRS (Toulouse, France). He has been working for more than 8 years on aerial robotics systems, carrying out research activities with the goal of providing aerial robots with the highest level of autonomy without the need for human intervention. His research interests can be summarized in three main topics: (1) perception and situation awareness based on sensor fusion, state estimation, localization and mapping, computer vision, and machine learning; (2) intelligent and cognitive system architectures for multi-agent systems; and (3) trajectory and path planning and control. He has authored more than 50 scientific publications related to these fields with near 900 citations and an h-index of 24 (source Google Scholar).

Manuel Castillo-Lopez is a PhD student at the Automation & Robotics Research Group of the Interdisciplinary Centre for Security Reliability and Trust (SnT) of the University of Luxembourg since September 2017. He received his MSc. in Industrial Engineering in 2017 at the University of Malaga. He has been working for 3 years on real-time motion planning and control of aerial robots with the goal of performing collision-free optimal trajectories with safety guarantees. His research interests span from optimization and probability theory, focusing on real-time stochastic optimal control.

Prof. Dr Miguel Olivares-Mendez is currently Assistant Professor on Space Robotics and Senior Research Scientist at the SnT-University of Luxembourg. He is also the head of the Space Robotics Research Group (SpaceR) and the Space Robotics & Automation Lab. He received the Diploma in Computer Science Engineering in 2006 (University of Malaga, UMA, Spain). He received the M.Sc. degree in Robotics and Automation and the PhD degree in Robotics and Automation (Technical University of Madrid, UPM, Spain), in 2009 and 2013, respectively. He got the Best PhD Thesis award of 2013 by the European Society for Fuzzy Logic and Technology (EUSFLAT). In May 2013 he joined the Interdisciplinary Center for Security Reliability and Trust (SnT) at the University of Luxembourg (Uni.Lu), as Associate Researcher in the Automation & Robotics Research Group. In December 2016 he became Research Scientist and main responsible of the research activities on mobiles robotics in the Automation & Robotics Research Group at the SnT-University of Luxembourg.

His main research interests are on UAVs, planetary and orbital robotics, computer vision, sensor fusion, vision-based control, soft-computing control techniques and robotics. He has published over 70 book chapters, scientific journals and conferences papers.

Holger Voos studied Electrical Engineering at the Saarland University and received the Doctoral Degree in Automatic Control from the Technical University of Kaiserslautern, Germany, in 2002. From 2000 to 2004, he was with Bodenseewerk Gertetechnik GmbH, Germany, where he worked as a Systems Engineer in aerospace and robotics. From 2004 to 2010, he was a Professor at the University of Applied Sciences Ravensburg-Weingarten, Germany, and the head of the Mobile Robotics Lab there. Since 2010, he is a Full Professor at the University of Luxembourg in the Interdisciplinary Centre for Security, Reliability and Trust (SnT), and the head of the SnT Automation and Robotics Research Group. His research interests are in the area of perception and control for autonomous vehicles and robots as well as distributed and networked control and automation.