2018 International Conference on Unmanned Aircraft Systems (ICUAS)
Dallas, TX, USA, June 12-15, 2018

Towards trajectory planning from a given path for multirotor aerial robots trajectory tracking

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

***Abstract*— Planning feasible trajectories given desired collision-free paths is an essential capability of multirotor aerial robots that enables the trajectory tracking task, in contrast to path following. This paper presents a trajectory planner for multirotor aerial robots carefully designed considering the requirements of real applications such as aerial inspection or package delivery, unlike other research works that focus on aggressive maneuvering. Our planned trajectory is formed by a set of polynomials of two kinds, acceleration/deceleration and constant velocity. The trajectory planning is carried out by means of an optimization that minimizes the trajectory tracking time, applying some typical constraints as m-continuity or limits on velocity, acceleration and jerk, but also the maximum distance between the trajectory and the given path. Our trajectory planner has been tested in real flights with a big and heavy aerial platform such the one that would be used in a real operation. Our experiments demonstrate that the proposed trajectory planner is suitable for real applications and it is positively influencing the controller for the trajectory tracking task.**

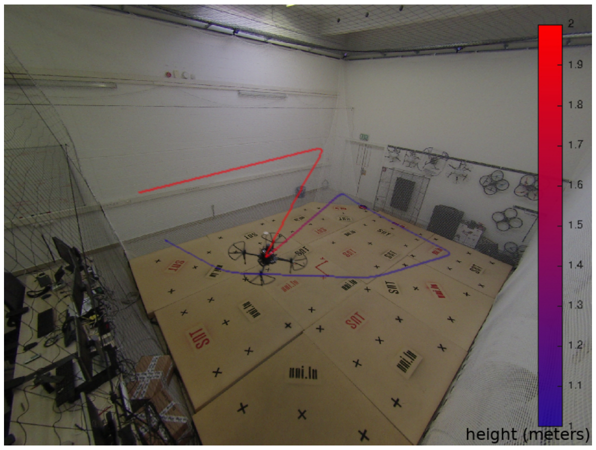


Fig. 1. Our aerial robot tracking the trajectory presented in Sect. VI.

I. INTRODUCTION

A. Motivation

During the last years, the use of multirotor aerial robots in many different applications, such as aerial inspection or package delivery, has became a reality. Recent advances in fully autonomous architectures and software frameworks like Aerostack [1], [2], have demonstrated the high-level of autonomy required to safely integrate aerial robots in daily use. Nevertheless, there are still multiple open research problems that limit the use of aerial robots. One of them is planning and tracking collision-free trajectories (see Fig. 1).

To solve the problem of planning and tracking collision-free trajectories, an approach frequently seen in the literature, [3], [4], [5], [6], is the combination of three components: (1) a collision-free geometric path planner, (2) a trajectory planner that computes a feasible trajectory from the previously given path, considering the limits on the dynamic of the robot, and (3) a controller for the trajectory tracking.

In this work we focus on planning feasible trajectories from given paths. These given paths might be provided by the user or by a geometric path planner such as [3], [7]. The maximum distance between the planned trajectories and the given paths must be bounded to avoid collisions with the obstacles of the environment, as a requirement for industrial oriented applications. Moreover, these kinds of applications demand sufficient payload of the aerial platforms to carry sensors and computers onboard, what imposes additional requirements on the trajectory planning. Finally, these trajectories are used as reference by the trajectory tracking controller.

B. State of the art

The field of trajectory generation from a given path is a recurrent topic found in the literature. There are multiple works related to robotic manipulators: In [8], 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 [9], [10], [11], 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.

The works that are specifically designed for aerial robots can be divided into (1) path smoothing and (2) trajectory planning. In the first group, in [12], the authors propose a kinematically feasible path for fixed-wing aerial robots that connects a series of waypoints, whereas a continuous curvature path smoothing for fixed-wing aerial robots is

This work was supported by the "Fonds National de la Recherche" (FNR), Luxembourg, under the project C15/15/10484117 (BEST-RPAS).

[1] Dr. Jose Luis Sanchez-Lopez, Dr. Miguel A. Olivares-Mendez, Manuel Castillo and Prof. Dr.-Ing. Holger Voos are with Automation and Robotics Research Group (ARG), Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. Address: 29, avenue J. F. Kenedy, L-1855 Luxembourg (Luxembourg). e-mail: joseluis.sanchezlopez@uni.lu

[2] Prof. Dr.-Ing. Holger Voos is with the Faculté des Sciences, de la Technologie et de la Communication, University of Luxembourg. e-mail: holger.voos@uni.lu

presented in [13]. In [5], [14], the authors propose a splines-based optimization to plan a path for multirotor aerial robots. In [14] the path is planned over a grid, whereas in [5], the path is planned with the help of Voronoi diagrams. The main drawback of these approaches is that 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.

To guarantee dynamic constraints of multirotor aerial robots, a full trajectory planning is needed. This can be done together with the control, such as in [15], [16], where authors use a model predictive controller (MPC) to track a path, but considering the dynamic model of the aerial robot, imposing a jerk-continuous movement. The main drawback of these approaches is the tightness between the trajectory planner and the controller. This enforces the need of re-planning in real-time which is impossible for long and complex trajectories.

Planning trajectories in a different component than the controller ensures a real-time execution. In [17], [18], [19], the authors propose a closed-form analytical trajectory planning approach for multirotor aerial robots. The trajectory has a multiple phases polynomial definition, as some other works mentioned above. In [18], [19], 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 or certain requirements of the waypoints.

Finally, in [6], [4], [20], 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 [6] incorporate a collision-free check, while [20] simply impose some distance to path limitations in certain discrete trajectory parts. In [6], [4], the snap is minimized, whereas in [20] 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 [4], [20], the time to pass for the waypoints is imposed externally and not computed by the planner.

C. Problem formulation

We assume to have a collision-free path, P, defined as a discrete set of waypoints that encode the desired position and heading of the aerial robot. The initial position of the robot is considered as the first waypoint.

The objective of the proposed trajectory planner is the generation of a trajectory, L, used by the trajectory tracking controller as reference. 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 follow the given path, without getting far away from it, and passing through the waypoints, and has to be feasible by the robot.

D. Contributions and outline

The first contribution of the paper is the definition of the trajectory as a set of two different kind of polynomials: acceleration/deceleration and constant velocity. This multi-phase trajectory definition allows to calculate, without lost of flexibility, an optimal solution but following a predefined simple profile. That is specially beneficial when tracking realistic trajectories for real applications such as inspection or package delivery, unlike other research-oriented aggressive maneuvers shown in the literature. It is important to highlight that our trajectory description allows though to impose continuity of the derivative of the pose at our choice.

Secondly, we developed a formulation of an optimization-based trajectory planning solution from a given path. Unlike analytical closed-form solution found in the literature, this formulation allows to easily modify the restrictions on the trajectory, including not only m-continuity of the trajectory, limits on velocity, acceleration and jerk, and constraints in the waypoints, but also, maximum distance between the trajectory and the given path. This last restriction is a must when intended to track realistic trajectories in cluttered environments for real applications, which is not incorporated in many works found in the literature, where these restrictions only apply in the waypoints.

Third, our approach avoids singularities on the representation of the orientation over the trajectory by using quaternions, unlike other state of art approaches that use Euler angles.

Finally, our trajectory minimizes the trajectory tracking time, instead of energy or snap, making it more suitable for the previously mentioned real applications. Moreover, in our experiments, we saw that minimizing the time, also reduces indirectly the energy and the snap.

The remainder of the paper is organized as follows: Sect. II includes the mathematical background needed in this work. Sect. III describes the aerial robot model, whereas Sect. IV defines the trajectory model used. In Sect. V, we state the optimization problem that the proposed trajectory planner has to solve. An evaluation and discussion of the results of this work is done in Sect. VI. Finally, Sect. VII concludes the paper and points out some future lines of work.

II. MATHEMATICAL BACKGROUND

A. Linear variables

The position of the robot in world coordinates is represented as $\boldsymbol{p} = [p_x,\ p_y,\ p_z]^T$, being the l-th derivative of the position of the robot with respect to world in world coordinates given by:

$$\boldsymbol{p}^{(l)} = \frac{\mathrm{d}^{(l)}\boldsymbol{p}}{\mathrm{d}t^{(l)}} = \begin{bmatrix} \frac{\mathrm{d}p_x^{(l)}}{\mathrm{d}t^{(l)}} \\ \frac{\mathrm{d}p_y^{(l)}}{\mathrm{d}t^{(l)}} \\ \frac{\mathrm{d}p_z^{(l)}}{\mathrm{d}t^{(l)}} \end{bmatrix}, \quad \forall l \in \mathbb{Z}, \quad l > 0 \qquad (1)$$

B. Angular variables

The orientation of the robot in world coordinates is represented by the unit quaternion, $\boldsymbol{q} = [q_w,\ \boldsymbol{q}_v] = [q_w,\ q_x,\ q_y,\ q_z]$, being the l-th derivative of the orientation of the robot with respect to world in world coordinates given by:

$$\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} = \frac{1}{2} \cdot \boldsymbol{\omega} \otimes \boldsymbol{q}, \quad l = 1 \tag{2}$$

$$\boldsymbol{\omega}^{(l-1)} = \frac{\mathrm{d}^{(l-1)}\boldsymbol{\omega}}{\mathrm{d}t^{(l-1)}} = \begin{bmatrix} \frac{\mathrm{d}\omega_x^{(l-1)}}{\mathrm{d}t^{(l-1)}} \\ \frac{\mathrm{d}\omega_y^{(l-1)}}{\mathrm{d}t^{(l-1)}} \\ \frac{\mathrm{d}\omega_z^{(l-1)}}{\mathrm{d}t^{(l-1)}} \end{bmatrix}, \quad \forall l \in \mathbb{Z}, \quad l > 1 \tag{3}$$

where in equation 2, abusing of notation, $\boldsymbol{\omega}$ represents the pure quaternion associated to the angular velocity $\boldsymbol{\omega}$, and $\otimes$ is the quaternion product.

The use of quaternions allows us to have a continuous and singularity-free representation of the orientation of the robot (unlike for instance Euler angles) with a reduced number of variables (unlike rotation matrices).

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

$$\delta\boldsymbol{q} = \boldsymbol{q}_a^* \otimes \boldsymbol{q}_b \tag{4}$$

where $\boldsymbol{q}_a^*$ is the conjugate of $\boldsymbol{q}_a$.

The error-quaternion, $\delta\boldsymbol{\theta}$, between two quaternions $\boldsymbol{q}_a$ and $\boldsymbol{q}_b$ is calculated as:

$$\delta\boldsymbol{\theta} = f_{\delta\boldsymbol{\theta}}\left(\boldsymbol{q}_a, \boldsymbol{q}_b\right) \approx 2 \cdot \delta\boldsymbol{q}_v \tag{5}$$

where $\delta\boldsymbol{q}_v$ is the vectorial part of the quaternion difference $\delta\boldsymbol{q}$. It requires that $\boldsymbol{q}_a \approx \boldsymbol{q}_b$, and therefore

$$\delta\boldsymbol{q} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix} \tag{6}$$

C. Angular variables: Simplification

In the case that we are only interested in the yaw angle, without considering the other two angles, the attitude of the aerial robot can be represented by the following simplified unit "s-quaternion", $\tilde{\boldsymbol{q}} = [q_w,\ q_z]$.

The l-th derivative of the orientation of the robot in world coordinates with respect to world can be simplified to:

$$\frac{\mathrm{d}\tilde{\boldsymbol{q}}}{\mathrm{d}t} = \frac{1}{2} \cdot \tilde{\boldsymbol{\omega}} \otimes \tilde{\boldsymbol{q}}, \quad l = 1 \tag{7}$$

$$\tilde{\boldsymbol{\omega}}^{(l-1)} = \frac{\mathrm{d}^{(l-1)}\tilde{\boldsymbol{\omega}}}{\mathrm{d}t^{(l-1)}} = \begin{bmatrix} \frac{\mathrm{d}\omega_\psi^{(l-1)}}{\mathrm{d}t^{(l-1)}} \end{bmatrix}, \quad \forall l \in \mathbb{Z}, \quad l > 1 \tag{8}$$

Using the algebra of the quaternions, simplified to the case of the "s-quaternion", and considering the unity property of the "s-quaternion", $|\tilde{\boldsymbol{q}}| = 1$, equation 7 is reduced to:

$$\begin{bmatrix} \frac{\mathrm{d}q_w}{\mathrm{d}t} \\ \frac{\mathrm{d}q_z}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \cdot \omega_\psi \cdot \sqrt{1 - q_w^2} \\ \frac{1}{2} \cdot \omega_\psi \cdot \sqrt{1 - q_z^2} \end{bmatrix} \tag{9}$$

After the integration of the previous differential equation, the relationship between the angular velocity with the orientation is simplified to:

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} \cos\left(\frac{1}{2} \cdot \rho_\psi\right) \\ \sin\left(\frac{1}{2} \cdot \rho_\psi\right) \end{bmatrix} \tag{10}$$

where the orientation-related value, ρ_ψ, is related to the angular velocity as

$$\omega_\psi = \frac{\mathrm{d}\rho_\psi}{\mathrm{d}t} \tag{11}$$

III. AERIAL ROBOT MODEL

According to [4], [21], the dynamics of an underactuated multirotor aerial robot is differentially flat. This means that the states and the inputs can be written as algebraic functions of four carefully selected flat outputs and their derivatives. This facilitates the generation of trajectories since any smooth trajectory (with reasonably bounded derivatives) in the space of flat outputs can be followed by the aerial robot.

A common choice of the flat outputs, [4], [21], are the position of the center of mass of the aerial robot in world coordinates, $\boldsymbol{p}_R^W = \left[p_x,\ p_y,\ p_z\right]^T$, and its yaw (heading) angle, that in this work we represent as a simplified quaternion $\tilde{\boldsymbol{q}}_R^W = [q_w,\ q_z]$.

The restrictions we have applied to our model are the following[2]:

- Bounded linear velocity, $\boldsymbol{v}_{R|W}^R$.
- Bounded angular velocity, $\omega_\psi{}_{R|W}^R$.
- Bounded linear acceleration, $\boldsymbol{a}_{R|W}^R$.
- Bounded angular acceleration, $\alpha_\psi{}_{R|W}^R$.
- Bounded linear jerk, $\boldsymbol{j}_{R|W}^R$.
- Bounded angular jerk, $\iota_\psi{}_{R|W}^R$.

We have seen that bounding higher-order derivatives is not needed to accurately represent the robot model for the calculation of a smooth trajectory.

IV. TRAJECTORY DEFINITION

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

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

Carrying out the following change on the time variable:

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

the trajectory, L, is redefined as:

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

The number of segments, n_s, of the trajectory, L, is related to the number of waypoints, n_w, of the given path, P. We propose to have three segments in the trajectory 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:

$$n_s = 3 \cdot (n_w - 1) \tag{12}$$

[2]The notation $\nu_{A|B}^C$ represents the value of the magnitude ν of the system A, with respect to system B, represented in coordinates of the system C.

All the segments, $\tilde{\boldsymbol{s}}_i(\tau_i)$, are 4-dimensional, having one dimension per flat output, i.e.

$$\tilde{\boldsymbol{s}}_i(\tau_i) = \begin{bmatrix} p_{i,x}(\tau_i) \\ p_{i,y}(\tau_i) \\ p_{i,z}(\tau_i) \\ \rho_{i,\psi}(\tau_i) \end{bmatrix}, \quad \forall i = \{1..n_s\} \tag{13}$$

.

The segments of every dimension $j = \{x, y, z, \psi\}$ are defined as polynomials functions of order m, i.e. the $(m+1)$-th derivative of the position, p_j, and the orientation-related value, ρ_ψ, is zero:

$$p_{i,j}(\boldsymbol{b}_{i,j,:}, \tau_i) = \sum_{k=0}^{k=m} \left(b_{i,j,k} \cdot \tau_i^k\right), \quad j = \{x, y, z\} \tag{14}$$

$$\rho_{i,\psi}(\boldsymbol{b}_{i,\psi,:}, \tau_i) = \sum_{k=0}^{k=m} \left(b_{i,\psi,k} \cdot \tau_i^k\right) \tag{15}$$

The value of the orientation, $\tilde{\boldsymbol{q}}_{i,\psi}(\boldsymbol{b}_{i,\psi,:}, \tau_i)$ is calculated combining equations 10 and 15.

The derivatives of the linear variables are calculated by differentiation of equation 14 as shown in equation 1. The first derivative of the angular variable requires the differentiation of equation 15 following equation 11. The higher-order derivatives of the angular variable are calculated following equation 8.

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

Although our presented solution is general enough, we propose a trajectory that is continuous up to third order, $n_d = 3$, that is, the third derivative, i.e. the jerk, is continuous but not derivable.

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, $m = 6$, with 7 coefficients per dimension and segment. These segments are used to represent 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, $m = 1$, with 2 coefficients per dimension and segment. These segments are used to represent a constant velocity movement of the robot.

V. TRAJECTORY PLANNER

The trajectory planner calculates the optimum trajectory, L^*, defined by the parameters, $\boldsymbol{b}^*$, and $\Delta\boldsymbol{\tau}^*$, as illustrated in Sect. IV, by solving the following single-objective nonlinear multivariable constrained minimization problem:

$$\boldsymbol{b}^*, \Delta\boldsymbol{\tau}^* = \underset{\boldsymbol{b}\in B, \Delta\boldsymbol{\tau}\in T}{\arg\min} \left(J(\Delta\boldsymbol{\tau})\right)$$

subject to:

Time feasibility:	$-\Delta\boldsymbol{\tau} \leq 0$
Derivability of the trajectory:	$c_s(\boldsymbol{b}, \Delta\boldsymbol{\tau}) = 0$
Dynamics of the robot:	$\nu_R(\boldsymbol{b}, \Delta\boldsymbol{\tau}) - \nu_{max} \leq 0$
	$\nu_{min} - \nu_R(\boldsymbol{b}, \Delta\boldsymbol{\tau}) \leq 0$
Waypoints:	$w(\boldsymbol{b}, \Delta\boldsymbol{\tau}, P) = 0$
Distance to path:	$d(\boldsymbol{b}, \Delta\boldsymbol{\tau}, P) - d_{max} \leq 0$

The proposed optimization problem is detailed in the following Sects. V-A to V-D.

A. Optimization variables

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

- The coefficients of the polynomials of the segments, $\boldsymbol{b} = \{b_{i,j,k}\} \in B$, $\forall i = \{1..n_s\}$, $\forall j = \{x, y, z, \psi\}$, and $\forall k = \{1..7\}$ for the case of waypoint connected polynomials, and $\forall k = \{1..2\}$ 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, $n_x = 67\cdot(n_w - 1)$. All the unknowns are real numbers, i.e. $\boldsymbol{x} \in \mathbb{R}^{n_x}$.

B. Objective function

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

$$J(\Delta\boldsymbol{\tau}) = \sum_{\forall i} \Delta\tau_i \tag{16}$$

C. Constraints

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

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

$$\Delta\tau_i \geq 0, \quad \forall i = \{1..n_s\} \tag{17}$$

2) Derivability of the trajectory: As mentioned in Sect. IV, the trajectory has to be derivable up to n_d-th order, i.e. T is 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. We represent this set of restrictions as $c_s(\boldsymbol{b}, \Delta\boldsymbol{\tau}) = 0$.

For the linear variables, we have $\forall i = \{1..n_s\}$, $\forall j = \{x, y, z\}$, and $\forall l = \{0..n_d\}$, the following expression:

$$p_{i,j}^{(l)}\left(\boldsymbol{b}_{i,j,:}, \Delta\tau_i\right) - p_{i,j}^{(l)}\left(\boldsymbol{b}_{i+1,j,:}, 0\right) = 0 \tag{18}$$

For the orientation, we have $\forall i = \{1..n_s\}$:

$$f_{\delta\tilde{\boldsymbol{\theta}}}\left(\tilde{\boldsymbol{q}}_{i,\psi}\left(\boldsymbol{b}_{i,\psi,:}, \Delta\tau_i\right), \tilde{\boldsymbol{q}}_{i,\psi}\left(\boldsymbol{b}_{i+1,\psi,:}, 0\right)\right) = 0 \tag{19}$$

where $f_{\delta\tilde{\boldsymbol{\theta}}}$ is the error-quaternion calculated following equation 5.

For the derivatives of the angular variables. we have $\forall i = \{1..n_s\}$, and $\forall l = \{1..n_d\}$:

$$\omega_{i,\psi}^{(l-1)}\left(\boldsymbol{b}_{i,\psi,:}, \Delta\tau_i\right) - \omega_{i,\psi}^{(l-1)}\left(\boldsymbol{b}_{i+1,\psi,:}, 0\right) = 0 \quad (20)$$

3) Dynamics of the robot: The trajectory has to fulfill all the constraints on the dynamics of the robot, presented in Sect. III, as

$$\nu_{min} \leq \nu_R(\boldsymbol{b}, \Delta\boldsymbol{\tau}) \leq \nu_{max} \quad (21)$$

4) Waypoints: The trajectory has to pass through the waypoints of the given path, P. We represent this set of restrictions as $w(\boldsymbol{b}, \Delta\boldsymbol{\tau}, P) = 0$.

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

$$p_{i_+,j}\left(\boldsymbol{b}_{i_+,j,:}, 0\right) - p_{w_n} = 0 \quad (22)$$

$$p_{i_-,j}\left(\boldsymbol{b}_{i_-,j,:}, \Delta\tau_{i_-}\right) - p_{w_n} = 0 \quad (23)$$

Similarly, for the orientation, we have:

$$f_{\delta\tilde{\boldsymbol{\theta}}}\left(\tilde{\boldsymbol{q}}_{i_+,\psi}\left(\boldsymbol{b}_{i_+,j,:}, 0\right), \tilde{\boldsymbol{q}}_{w_n}\right) = 0 \quad (24)$$

$$f_{\delta\tilde{\boldsymbol{\theta}}}\left(\tilde{\boldsymbol{q}}_{i_-,\psi}\left(\boldsymbol{b}_{i_-,j,:}, \Delta\tau_{i_-}\right), \tilde{\boldsymbol{q}}_{w_n}\right) = 0 \quad (25)$$

where $f_{\delta\tilde{\boldsymbol{\theta}}}$ is the error-quaternion calculated following equation 5.

The reader must note that the waypoint n is characterized by is position, p_{w_n} and orientation, $\tilde{\boldsymbol{q}}_{w_n}$.

5) Distance to path: The euclidean distance between the given path and the position variables of the trajectory has to be lower than a value, d_{max}. We represent this restriction as $d(\boldsymbol{b}, \Delta\boldsymbol{\tau}, P) \leq d_{max}$, and it is calculated as:

$$\|\boldsymbol{p}_{i,:}\left(\boldsymbol{b}_{i,xyz,:}, \tau\right) - \boldsymbol{p}_{P_{n,n+1}}\|_2 \leq d_{max} \quad (26)$$

$\forall\tau \in [0, \Delta\tau_i]$, $\forall i = \{1..n_s\}$, and for all the position subpaths, $\boldsymbol{p}_{P_{n,n+1}}$, that form the complete path P.

D. Initialization

The initialization of the unknowns is essential for a fast convergence of the optimization algorithm to a local minima. The initial value of the unknowns, $\boldsymbol{x}_0$, has to be feasible from the optimization problem point of view, i.e. it has to fulfill the constraints described in Sect. V-C.

We propose an initial trajectory, L_0, that exactly follows the given path, P, by setting to zero the velocity and all the higher order derivatives in the waypoints. This means that, between every two waypoints, three stages take place: acceleration from zero velocity, constant velocity movement, and deceleration to zero velocity. The maximum jerk, the maximum acceleration and the maximum velocity of this trajectory is given by the constraints of the dynamics of the robot. Our initial trajectory also follows its derivability constraints.

The aforementioned initial trajectory, L_0, is 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.

VI. EVALUATION AND RESULTS

A. Evaluation methodology

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

In the one hand, in Sect. VI-B, we evaluate the proposed trajectory planner in terms of the optimization process presented in Sect. V, and the properties of the calculated trajectory. We use the root mean square (RMS) value, ν_{rms}, of different magnitudes, ν, of the trajectory:

$$\nu_{rms} = \sqrt{\frac{1}{t_f} \cdot \int_0^{t_f} \nu^2(t) \cdot \mathrm{d}t}$$

In the other hand, the usefulness of the generated trajectory when used together with a controller is evaluated in Sect. VI-D by means of real flights. The experimental setup of the real flights is described in Sect. VI-C. It is important to highlight that this paper focuses on trajectory planning, and therefore, the performance of the control solution used in these experiments is not quantitatively evaluated.

We have implemented our trajectory planner in MATLAB, using the single-objective nonlinear multivariable constrained minimization solver provided by the function $fmincon$.

B. Trajectory planning results

In this section, we evaluate the proposed trajectory planner, analyzing the optimization process and the properties of the calculated trajectory. The path used for the evaluation is defined by the 9 waypoints listed in Table I and it is shown in Fig. 2. We assume the aerial robot to be initially hovering with a pose that coincides with the first waypoint. Despite being a short path[3], it is very challenging, as it includes simultaneous movements in the four degrees of freedom of the aerial robot. It is important to highlight that we have artificially included waypoint 5 for evaluation purposes.

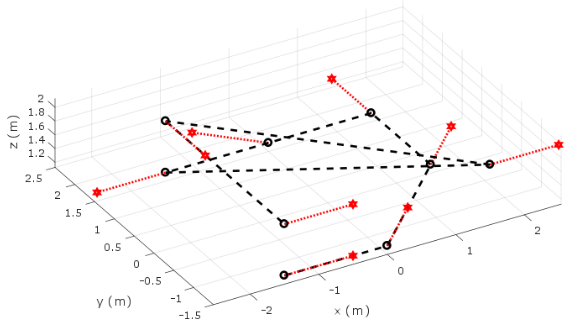


Fig. 2. 3D view of the path used for the evaluation. The path is represented with a dashed black line, being its waypoints, listed in Table I, represented by a circle (position) and a red arrow (heading).

We have configured our trajectory planner with the configuration parameters listed in Table II.

[3]Note that the length of the path is limited by the size of our flight arena.

	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9
p_x (m)	-2	0	2	2	0	-2	2	-2	-2
p_y (m)	-2	-2	0	2	2	2	-2	2	-2
p_z (m)	1.25	1.25	1.25	1.25	1.25	1.25	2	2	2
ψ (°)	0	45	45	90	135	180	0	-90	0

TABLE I

LIST OF WAYPOINTS OF THE PATH USED FOR THE EVALUATION.

	d_{path} (m)	${v}^{W}_{R\|W}$ (m·s^{-1})	${\omega_\psi}^{W}_{R\|W}$ (rad·s^{-1})	${a}^{W}_{R\|W}$ (m·s^{-2})	${\alpha_\psi}^{W}_{R\|W}$ (rad·s^{-2})	${j}^{W}_{R\|W}$ (m·s^{-3})	${\iota_\psi}^{W}_{R\|W}$ (rad·s^{-3})
min	0	-1.5	-1.5	-2	-2	-5	-5
max	0.05	1.5	1.5	2	2	5	5

TABLE II

CONFIGURATION PARAMETERS OF THE TRAJECTORY PLANNER.

As described in Sect. V-D, we initialize the optimization problem that our trajectory planner has to solve with an initial trajectory that fulfills all the constraints and exactly follows the given path. The values (pose and derivatives up to jerk) of this initial trajectory are plotted in Fig. 3 and Fig. 4. The total time of the trajectory tracking of this initial trajectory is $t = 24.35$ s.

Fig. 5 shows the 3D view of the calculated trajectory once the optimization process ended. The values (pose and derivatives up to jerk) of this trajectory are plotted in Fig. 6 and Fig. 7. The total time of the trajectory tracking of this trajectory is $t = 18.1$ s.

As expected, the calculated trajectory still fulfills all the constraints, but the total time of the trajectory tracking has decreased by 25.67% from the initial trajectory ($t = 24.35$ s.) to the optimized one ($t = 18.1$ s.). Considering the initial trajectory without waypoint 5, the total time of the trajectory tracking has decreased by 21.13% (from $t = 22.95$ s.).

The reader is recommended to deeply compare Fig. 3 and Fig. 4 with Fig. 6 and Fig. 7 to perceive the difference between the two trajectories. For example, in the artificial waypoint 5, the velocity in x-axis of the initial trajectory was 0 m·s^{-1}, while in the optimized trajectory is -1.5 m·s^{-1}, as we would not have included it as a waypoint of the path. In the rest of the waypoints, the velocity has also increased.

Moreover, as shown in Table III, the RMS value of the velocity of the trajectory has increased by more than 10% while the RMS of the other higher order derivatives has decreased (except the linear acceleration). This means that the calculated trajectory is not only faster than the initial one, but also smoother and more energy-efficient.

	$\boldsymbol{v}$	ω_ψ	$\boldsymbol{a}$	α_ψ	$\boldsymbol{j}$	ι_ψ	$\boldsymbol{s}$	σ_ψ
initial (a)	0.9853	0.5169	1.1665	0.5867	3.1034	1.5701	15.6377	7.9575
initial w/o W_5 (b)	1.0336	0.5422	1.1487	0.5754	2.9555	1.4862	14.3780	7.2615
optimized (c)	1.1783	0.6122	1.2062	0.5309	2.7346	1.1411	13.8708	5.5012
% change (a)-(c)	19.58	18.43	3.40	-9.50	-11.88	-27.32	-11.30	-30.87
% change (b)-(c)	14.00	12.90	5.01	-7.73	-7.47	-23.22	-3.53	-24.24

TABLE III

RMS VALUES OF DIFFERENT MAGNITUDES OF BOTH INITIAL AND OPTIMIZED TRAJECTORIES.

As expected, the optimization process eventually converges to a feasible solution that minimizes the total time of trajectory tracking. In case of real-time constraints and / or limited computational resources availability, the user would be able to stop the optimization process at any iteration. In such case, the calculated trajectory would still be feasible (i.e. all the constraints would be fulfilled), but it would not have converged to a minimum of the total time of trajectory tracking. Nevertheless, this solution would be a better choice than the initial trajectory, in terms of the RMS values discussed before.

C. Experimental setup

In Sect. VI-D, we evaluate the usefulness of the generated trajectory when used together with a controller by means of real flights (see 1), describing along this section the experimental setup.

Fig. 8 shows the proposed system architecture for the experimental evaluation. All the components, except the trajectory planner, have been implemented in C++ using ROS [22] as middleware.

Our aerial robot platform is a DJI Matrice 100[4] quadrotor (see Fig. 9). 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 (including a DJI Guidance[5]). 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 [4], [21], 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 for industrial oriented applications.

We model our platform (mechanical part together with the autopilot with the velocity controller) following a similar procedure as described in [23]. We define the state of the aerial robot platform as:

$$\boldsymbol{x} = \left[\boldsymbol{p}^{W}_{R},\ \tilde{\boldsymbol{q}}^{W}_{R},\ \boldsymbol{v}^{R}_{R|W},\ {\omega_\psi}^{R}_{R|W}\right]^{T} \tag{27}$$

And the state-space model, $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$, that represents the dynamics of the system, is as follows:

$$\dot{\boldsymbol{p}}^{W}_{R} = R^{W}_{R} \cdot \boldsymbol{v}^{R}_{R|W} \tag{28}$$

$$\dot{v_j}^{R}_{R|W} = -\frac{1}{\tau_j} \cdot {v_j}^{R}_{R|W} + \frac{k_j}{\tau_j} \cdot u_j, \quad \forall j = \{x, y, z\} \tag{29}$$

$$\dot{\tilde{\boldsymbol{q}}}^{W}_{R} = \frac{1}{2} \cdot {\omega_\psi}^{R}_{R|W} \otimes \tilde{\boldsymbol{q}}^{W}_{R} \tag{30}$$

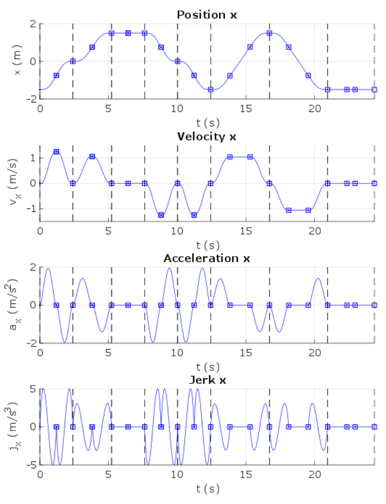
$$\dot{\omega_\psi}^{R}_{R|W} = -\frac{1}{\tau_\psi} \cdot {\omega_\psi}^{R}_{R|W} + \frac{k_\psi}{\tau_\psi} \cdot u_\psi \tag{31}$$

where R^{W}_{R} is the rotation matrix associated to the simplified quaternion $\tilde{\boldsymbol{q}}^{W}_{R}$. The values of the parameters k_j and τ_j have been calculated empirically, and are shown in Table IV. Note that in this particular case, ${\omega_\psi}^{R}_{R|W} = {\omega_\psi}^{W}_{R|W}$, as we are only considering the heading angle.

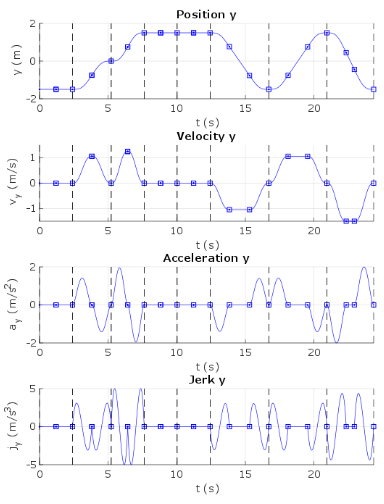
We collect information about the state of our aerial platform by means of an Optitrack motion capture system, which

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

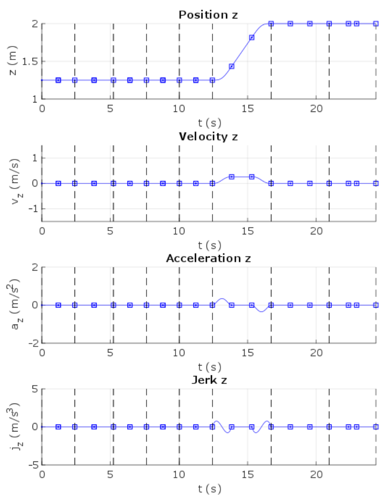
[5]https://www.dji.com/guidance



(a) x.



(b) y.



(c) z.

Fig. 3. Initial 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 time of tracking is $t = 24.35$ s.

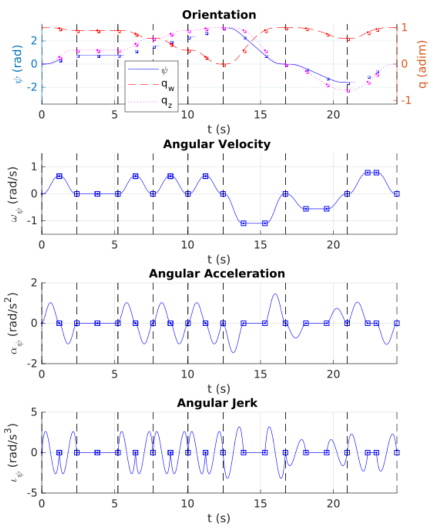


Fig. 4. Initial trajectory. Heading, and its derivatives up to jerk, plotted in dashed red and magenta and in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total time of tracking is $t = 24.35$ s.

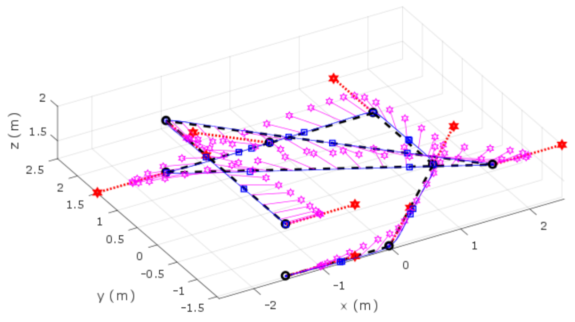


Fig. 5. 3D view of the calculated trajectory. The position values are drawn with a solid blue line, whereas the heading is represented with magenta arrows. The path is displayed with a dashed black line, being its waypoints represented by a circle (position) and a red arrow (heading).

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

TABLE IV
EMPIRICALLY CALCULATED VALUES OF THE DYNAMICS OF OUR AERIAL PLATFORM.

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 [24].

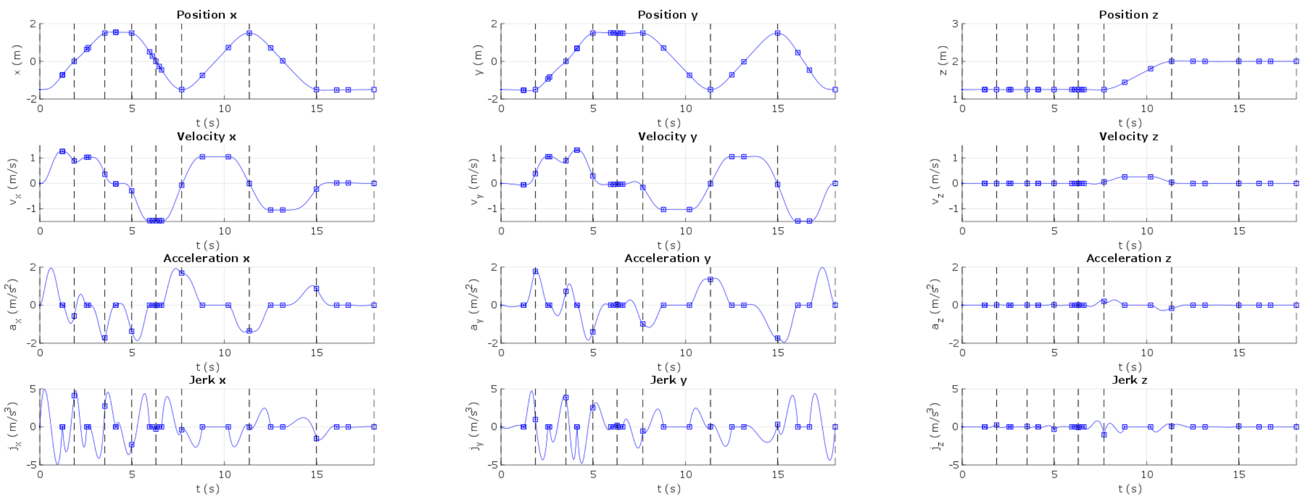
Our control loop is formed by a feedforward controller and a feedback controller in parallel, $\boldsymbol{u} = \boldsymbol{u}_{ff} + \boldsymbol{u}_{fb}$, as shown in Fig. 8. We set the frequency of the control loop to 50 Hz.

The feedforward controller inverts the model of the robot and feeds it with the desired values of the magnitude of the planned trajectory to calculate the control commands, $\boldsymbol{u}_{ff} = f^{-1}(\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d)$, as:

$$u_{ff,j} = \frac{\tau_j}{k_j} \cdot a_{d,j}{}^{R}_{R|W} + \frac{1}{k_j} \cdot v_{d,j}{}^{R}_{R|W}, \quad \forall j = \{x, y, z\} \tag{32}$$

$$u_{ff,\psi} = \frac{\tau_\psi}{k_\psi} \cdot \alpha_{d,\psi}{}^{R}_{R|W} + \frac{1}{k_\psi} \cdot \omega_{d,\psi}{}^{R}_{R|W} \tag{33}$$

where, as mentioned before, $\boldsymbol{v}_d{}^{R}_{R|W} = (R_d{}^{W}_{R})^T \cdot \boldsymbol{v}_d{}^{W}_{R|W}$,



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

Fig. 6. Calculated 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 time of tracking is $t = 18.1$ s.

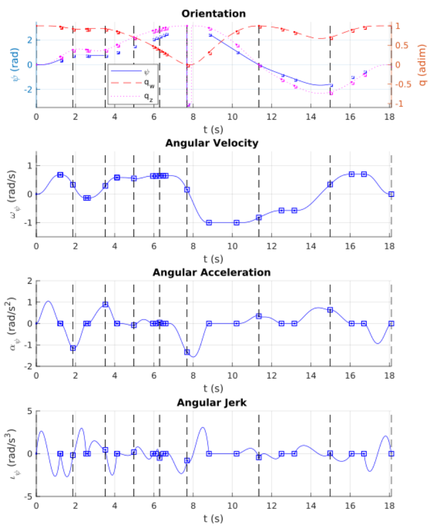


Fig. 7. Calculated trajectory. Heading, and its derivatives up to jerk, plotted in dashed red and magenta and in solid blue. The blue dots and squares illustrate the beginning and end of a segment of the trajectory. The vertical dashed black lines represent the waypoints. Its total time of tracking is $t = 18.1$ s.

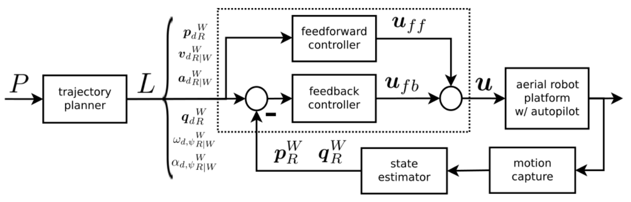

P
trajectory planner
L
$\boldsymbol{p}_{dR}^{W}$
$\boldsymbol{v}_{d\,R|W}^{W}$
$\boldsymbol{a}_{d\,R|W}^{W}$
$\boldsymbol{q}_{dR}^{W}$
$\omega_{d,\psi\,R|W}^{W}$
$\alpha_{d,\psi\,R|W}^{W}$
feedforward controller
$\boldsymbol{u}_{ff}$
feedback controller
$\boldsymbol{u}_{fb}$
$\boldsymbol{u}$
-
aerial robot platform w/ autopilot
$\boldsymbol{p}_{R}^{W}$ $\boldsymbol{q}_{R}^{W}$
state estimator
motion capture

Fig. 8. System architecture setup for the experimental evaluation of the proposed trajectory planner.

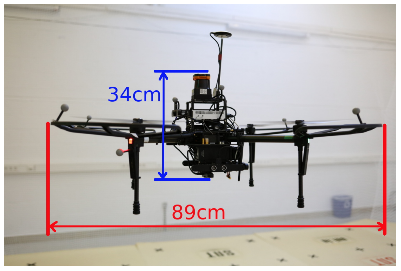


Fig. 9. The aerial robot platform used for the experimental validation.

$\boldsymbol{a}_{d\,R|W}^{\;R} = (R_{dR}^{W})^{T} \cdot \boldsymbol{a}_{d\,R|W}^{\;W}$, $\alpha_{d,\psi\,R|W}^{\;R} = \alpha_{d,\psi\,R|W}^{\;W}$ and $\omega_{d,\psi\,R|W}^{\;R} = \omega_{d,\psi\,R|W}^{\;W}$.

The feedback controller is based on [25], and it is formed by four fuzzy logic based controllers that calculate the control command considering the desired pose of the platform given by the trajectory planner and the estimated current pose of the robot:

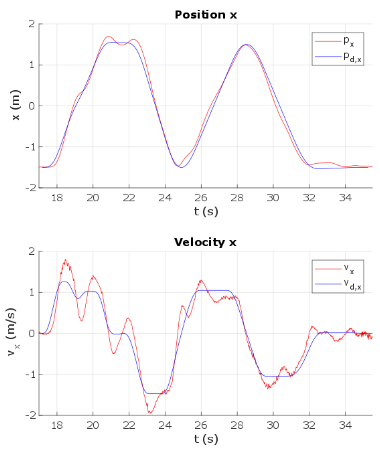
$$\boldsymbol{u}_{fb} = \boldsymbol{c}_{FL}\left(\boldsymbol{p}_{R}^{W}, \tilde{\boldsymbol{q}}_{R}^{W}, \boldsymbol{p}_{dR}^{W}, \tilde{\boldsymbol{q}}_{dR}^{W}\right) \tag{34}$$

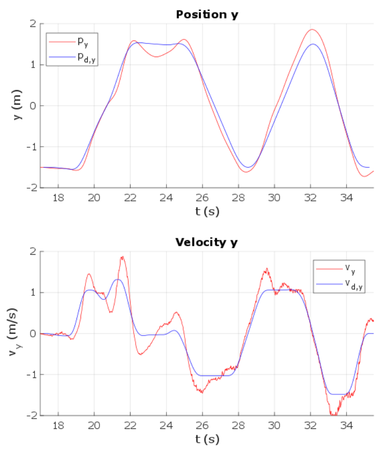
Finally, as mentioned before, the trajectory planner has been implemented in MATLAB and it is executed offline.

D. *Experimental results*

Fig. 10 and Fig. 11 show the desired values of the position, orientation and velocity (in blue) of the calculated trajectory presented in Sect. VI-B, and the estimated values of the same magnitudes of the robot during the presented experiment (in red).

As can be seen, the trajectory is properly tracked not only in position, but also in velocity. It is important to remind the reader that this work focuses on trajectory planning and not in control, and as stated before, the objective of this section is simply to demonstrate the usefulness of the trajectory planner


(a) x.


(b) y.

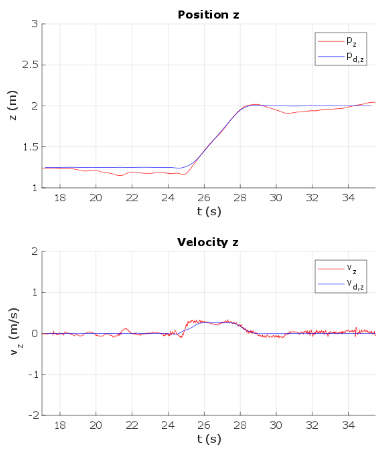

(c) z.

Fig. 10. Position and linear velocity of the robot during the experiment of tracking the trajectory calculated in Sect. VI-B. In blue the desired values calculated by the proposed trajectory planner. In red the estimated values that the robot had during the execution of the experiment.

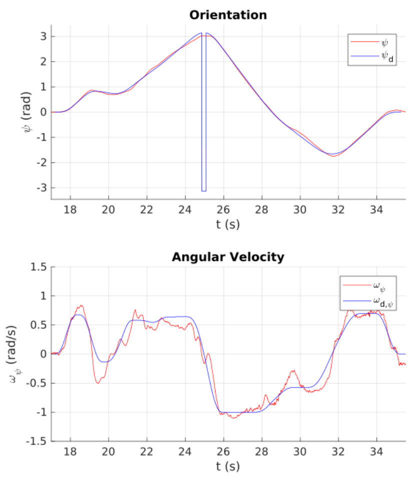
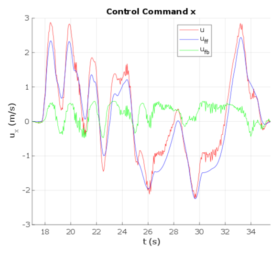
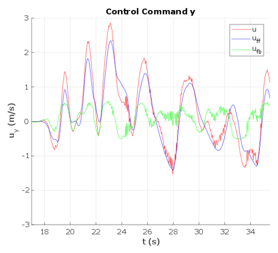
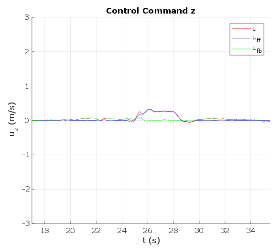


Fig. 11. Heading and angular velocity of the robot during the experiment of tracking the trajectory calculated in Sect. VI-B. In blue the desired values calculated by the proposed trajectory planner. In red the estimated values that the robot had during the execution of the experiment.


(a) u_x.


(b) u_y.


(c) u_z.

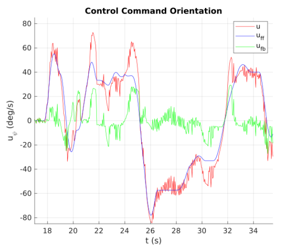

(d) u_ψ.

Fig. 12. Control commands given to the robot during the presented experiment to track the trajectory calculated in Sect. VI-B. The total control command (in red) is the addition of the feedforward command (in blue) and the feedback command (in green).

when used together with a controller in real flights, but we are not interested in analyzing quantitatively the error on the trajectory tracking. Therefore, we have used the control loop described in Sect. VI-C without an extra tuning or complex modeling.

Fig. 12 shows the control commands given to the robot to track the trajectory calculated in Sect. VI-B. The total control command (in red) is the addition of the feedforward command (in blue) and the feedback command (in green).

As can be extracted from Fig. 12, the feedforward controller is generating the major part of the total control command, reducing the effort of the feedback controller (and therefore making the trajectory tracking less dependent on it). It is, therefore, important to highlight the usefulness of feedforward controller despite the very simple model that we use to represent the dynamics of the aerial robot (and therefore to calculate the feedforward controller). Moreover, the reader might remember that the use of this feedforward controller is only possible thanks to the trajectory planner, which allows to have the desired values of the higher order derivatives that permits the inversion of the dynamic model of the robot.

Finally, the reader might note that our aerial robot (see Fig. 9) is considerably bigger and heavier (Size: $890 \times 890 \times 340$ mm^3, MTOW: 3600 g.) than the aerial platforms used in works like [6], [4], [20] that focus on aggressive maneuvering (e.g. AscTec Hummingbird[6]. Size: $540 \times 540 \times 85.5$ mm^3, MTOW: 710 g.).

VII. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an algorithm for multirotor aerial robots that calculates a feasible trajectory given a desired path. We have carefully designed our planner considering the requirements of real applications such as aerial inspection or package delivery, unlike other research works oriented to aggressive maneuvering. Our planned trajectory is formed by a set of polynomials of two kinds, acceleration/deceleration and constant velocity. The trajectory planning is carried out by means of an optimization that minimizes the trajectory tracking time, applying some typical constraints as m-continuity or limits on velocity, acceleration and jerk, but also the maximum distance between the trajectory and the given path.

We have tested our trajectory planner by means of real flights with a big and heavy aerial platform such the one that would be used in a real industrial application. In our experiments we have used a simple trajectory tracking control architecture that combines a feedforward and a feedback controller based on simple models and basic tuning. Our experiments have demonstrated that our trajectory planner is suitable for real applications and it is positively influencing the controller for the trajectory tracking task.

Some future work lines 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 variables in the waypoints (e.g. velocity in the waypoints). Very interesting would be to carry out a deep analysis of the initialization of the optimization and the configuration parameters. Finally, to completely solve the trajectory tracking problem, the control architecture is required to be improved.

REFERENCES

[1] J. L. Sanchez-Lopez, R. A. S. Fernández, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy, "Aerostack: An architecture and open-source software framework for aerial robotics," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 332–341.

[2] J. L. Sanchez-Lopez, M. Molina, H. Bavle, C. Sampedro, R. A. Suárez Fernández, and P. Campoy, "A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework," *Journal of Intelligent & Robotic Systems*, pp. 1–27, 2017.

[3] J. L. Sanchez-Lopez, J. Pestana, and P. Campoy, "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)*, June 2017, pp. 316–325.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.

[5] K. B. Judd and T. W. McLain, "Spline based path planning for unmanned air vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, vol. 9. Montreal, Canada, 2001.

[6] C. Richter, A. Bry, and N. Roy, *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*. Cham: Springer International Publishing, 2016, pp. 649–666.

[7] J. L. Sanchez-Lopez, M. Wang, M. A. Olivares-Mendez, M. Molina, and H. Voos, "A real-time 3d trajectory planning solution for collision-free navigation of multirotor aerial robots in dynamic environments," *Journal of Intelligent & Robotic Systems*, pp. 1–32, 2018.

[8] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2493–2498.

[9] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42–52, Feb 2003.

[10] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 3248–3253.

[11] B. Ezair, T. Tassa, and Z. Shiller, "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *The International Journal of Robotics Research*, vol. 33, no. 6, pp. 898–916, 2014.

[12] S. G. Manyam, S. Rathinam, D. Casbeer, and E. Garcia, "Tightly bounding the shortest dubins paths through a sequence of points," *Journal of Intelligent & Robotic Systems*, vol. 88, no. 2, pp. 495–511, Dec 2017.

[13] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, June 2010.

[14] D. Jung and P. Tsiotras, "On-line path generation for small unmanned aerial vehicles using b-spline path templates," in *AIAA Guidance, Navigation and Control Conference, AIAA*, vol. 7135, 2008.

[15] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *2013 European Control Conference (ECC)*, July 2013, pp. 1383–1389.

[16] M. W. Mueller, M. Hehn, and R. D'Andrea, "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*, Nov 2013, pp. 3480–3486.

[17] A. Boeuf, J. Corts, R. Alami, and T. Simon, "Planning agile motions for quadrotors in constrained environments," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 218–223.

[18] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016, pp. 87–96.

[19] S. Beul, M. Behnke, "Fast full state trajectory generation for multirotors," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2017, pp. 408–416.

[20] M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Inversion based direct position control and trajectory following for micro aerial vehicles," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2933–2939.

[21] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, April 2018.

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[23] M. Castillo-Lopez, M. A. Olivares-Mendez, and H. Voos, "Evasive maneuvering for uavs: An mpc approach," in *ROBOT 2017: Third Iberian Robotics Conference*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. Cham: Springer International Publishing, 2018, pp. 829–840.

[24] J. L. Sanchez-Lopez, V. Arellano-Quintana, M. Tognon, P. Campoy, and A. Franchi, "Visual marker based multi-sensor fusion state estimation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 16 003 – 16 008, 2017, 20th IFAC World Congress.

[25] M. Olivares-Mendez, S. Kannan, and H. Voos, "Vision based fuzzy control autonomous landing with uavs: From v-rep to real experiments," in *Control and Automation (MED), 2015 23th Mediterranean Conference on*, June 2015, pp. 14–21.

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