Journal of Intelligent & Robotic Systems (2020) 98:241–265
https://doi.org/10.1007/s10846-019-01085-z

A Survey of Path Following Control Strategies for UAVs Focused on Quadrotors

Bartomeu Rubí[1] · **Ramon Pérez**[1] · **Bernardo Morcego**[1]

Received: 9 May 2019 / Accepted: 22 August 2019 / Published online: 5 September 2019

Abstract

The trajectory control problem, defined as making a vehicle follow a pre-established path in space, can be solved by means of trajectory tracking or path following. In the trajectory tracking problem a timed reference position is tracked. The path following approach removes any time dependence of the problem, resulting in many advantages on the control performance and design. An exhaustive review of path following algorithms applied to quadrotor vehicles has been carried out, the most relevant are studied in this paper. Then, four of these algorithms have been implemented and compared in a quadrotor simulation platform: *Backstepping* and *Feedback Linearisation* control-oriented algorithms and *NLGL* and *Carrot-Chasing* geometric algorithms.

Keywords Unmanned aerial vehicles · Trajectory control · Path following · Backstepping · Feedback Linearization · NLGL · Carrot-Chasing

1 Introduction

In recent years, the growing interest to develop fully autonomous aerial vehicles, known as Unmanned Aerial Vehicles (UAVs), has seen a civil market demand increase compared to its military applications. The most innovative applications include interaction of the UAVs with their environment (e.g. infrastructure maintenance and parcel delivery) focusing in the control and disturbance rejection. Nevertheless, the current main applications are supervision and mapping. These observations include applications as: environmental protection, including floods, fire and other disasters; security; agriculture, both in water and plague management; infrastructure supervision, including pipes and electrical lines. The main advantages of the UAVs are the accessibility, security and economy of their observations. These interests drive a continuous progress both in the artificial vision and trajectory control areas [50].

Out of all UAVs, multirotors stand out for their good manoeuvrability, stability and payload. Initially, the object of study of these vehicles was on obtaining controllers capable of stabilizing their attitude, the fastest and most influential dynamics. The stabilization control problem for the particular case of a quadrotor has been solved using different techniques such as *Backstepping*, *Feedback Linearisation*, *Sliding Mode Control*, *PID*, optimal control, robust control, learning-based control, etc [10, 48, 64]. Given that the stabilization control has already been widely studied, today the challenge for quadrotors is trajectory control, fault tolerant control, path planning or obstacle avoidance. The trajectory control problem, defined as making a vehicle follow a pre-established path in space, can be solved mainly by two different approaches: using a trajectory tracking controller or with a path following controller. For the trajectory tracking problem a reference specified in time is tracked, where the references of the path

This work has been partially funded by the Spanish State Research Agency (AEI) and the European Regional Development Fund (ERFD) through the project SCAV (ref. MINECO DPI2017-88403-R). Bartomeu Rubí is also supported by the Secretaria d'Universitats i Recerca de la Generalitat de Catalunya, the European Social Fund (ESF) and the AGAUR under a FI grant (ref. 2017FI B 00212).

✉ Bartomeu Rubí
tomeu.rubi@upc.edu

Ramon Pérez
ramon.perez@upc.edu

Bernardo Morcego
bernardo.morcego@upc.edu

[1] Research Center for Supervision, Safety and Automatic Control (CS2AC), Universitat Politècnica de Catalunya (UPC), Rbla Sant Nebridi 22, Terrassa, Spain

are given by a temporal evolution of each space coordinate. Whereas path following (PF) handles the problem of following a path with no preassigned timing information, thus any time dependence of the problem is removed.

In [2] the authors demonstrate that following a geometric path is less demanding than tracking a timed reference signal. They argue that, although it is possible to perfectly track any reference with minimum phase stable systems, the tracking error increases in non-linear systems with presence of unstable zero dynamics as the signal frequencies approach those of the unstable zeros. PF controllers offer a number of advantages over trajectory tracking controllers, not only these are easier to design [42] but also result in smoother convergence to the path and less demand on the control effort [15], a smaller transient error and a stronger robustness [77] and the control signals are less likely to be saturated [25]. The path following problem has also applications in underwater and ground-based robots. In all of those systems, the PF controller usually operates after another control block known as the path planner. The path planner is a high-level controller whose mission is generally to obtain an optimal path between two given points in space. The resulting optimal path, which commonly seeks a compromise between a minim path length and a minimum control effort, will vary based on the problem requirements. Occasionally, between the path following and the path planning controllers the path smoothing block can be found [21, 40]. The task of this block is to smooth the path generated by the path planning algorithm considering the kinematic constrains of the vehicle. However, the smoothing task is typically integrated inside the path planner. Apart from that, in some cases the path planning controller is also responsible for the obstacle avoidance task [18, 22].

Even though trajectory tracking surveys applied to multi-rotors are available, such as [47], there are no path following reviews applied to these vehicles. With this in mind, the present survey of path following focused on quadrotors is necessary. This survey compares PF algorithms on the same scenario in equal conditions and harmonizes the nomenclature: it gives a general definition of the path follo wing problem and offer details of the studied algorithms.

A thorough review of literature related to PF control applied to quadrotors has been carried out. It is important to mention that some papers use the term path following when implementing controllers that track timed trajectory references. As these papers do not follow our path following definition, they are not included in our bibliographic review. The trajectory control problem, as opposed to the stabilization problem, is less dependent on the plant. Therefore, PF techniques not applied to quadrotors are also included and references for other type of UAVs are provided. From this study, four of these PF algorithms have been chosen to be implemented and compared. They were selected for their popularity and performance. These algorithms are described in detail and applied to a simulation model.

The rest of the paper is organized as follows: In Section 2 the Path Following problem is defined. Section 3 exposes a literature review of the Path Following control applied to quadrotor vehicles. In Section 4 the quadrotor dynamic model is presented and then *Backstepping* and *Feedback Linearisation* control-oriented algorithms, and NLGL and *Carrot-Chasing* geometric algorithms are implemented to solve the PF problem. Section 5 reports the simulation results of the studied algorithms, and discusses their performance. Finally, Section 6 details the conclusions.

2 Path Following Problem

The topic of this paper addresses the trajectory control problem of a quadrotor vehicle. This problem is defined as making a quadrotor follow a pre-established path in space. Only PF control strategies are reviewed in this paper. The PF problem [1, 15, 42] is defined as:

Definition 1 Path Following Problem: Let the desired path be described by a curve in the three-dimensional space $\mathbf{p}_d(\gamma) := [x_d(\gamma), y_d(\gamma), z_d(\gamma)]^T$, parametrized by the virtual arc $\gamma \in [0; \gamma_f]$, where γ_f is the total virtual arc length. The control objective is to design a control law for the vehicle that ensures convergence of the vehicle's position $\mathbf{p}(t)$ to the path $\mathbf{p}_d(\gamma)$.

It is important to mention that some control-oriented algorithms may need a timing law for the virtual arc parameter, $\gamma(t)$, since some of this techniques consist in an adaptation of a trajectory tracking algorithm, such as *Backstepping*.

There are several approaches to define the desired path. Dubins defines the path as a combination of circle's arcs and lines tangent to them [8, 28]. In the waypoint-based case the sequence of points is commonly connected by straight-lines [21, 60] or splines [27, 84]. The most generic approach is a continuous function parametrized by the virtual arc length [4, 15, 42].

The guidance and control system's design to solve the PF problem is done by applying two different methodologies: the Separated Guidance and Control (SGC) approach and the Integrated Guidance and Control (IGC) approach [23]. The SGC approach is based on a separation between translational dynamics and rigid-body rotational dynamics. It consists on an outer-loop guidance law for generation of the movement commands and an inner-loop controller to track those commands, sometimes known as the autopilot. While the IGC approach combines both controllers in

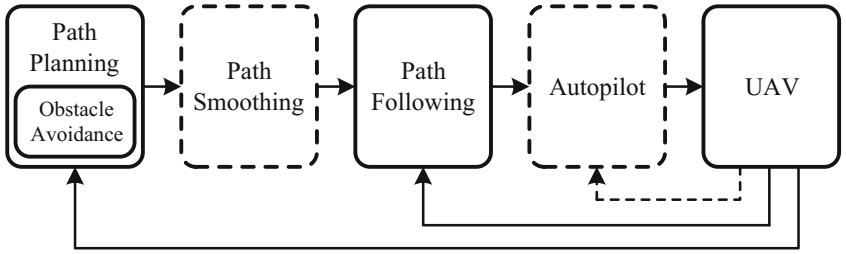

Fig. 1 Control structure

the same control loop. Figure 1 shows the typical UAVs guidance and control structure, which is a hierarchical control structure. The path smoothing block and the autopilot are represented with dashed lines as they are not always present in this control structure. As seen above, the path smoothing task can be integrated in the path planning block and in an IGC structure the path following block integrates the autopilot.

3 Path Following in Quadrotors

In this section the path following state of art applied on quadrotors is reviewed. We describe the control structures and algorithms most commonly used to solve the path following problem. The algorithms are organized in subsections, and a qualitative comparison is given. The algorithms are organized in subsections and a qualitative comparison is exposed. The path following problem has been widely studied for Ground Vehicles and fixed-wing aircrafts but its application to quadrotors is a relatively new research field. For that reason, references for other types of UAVs are reported when an algorithm is not found applied to quadrotors.

3.1 Backstepping

Backstepping (*BS*) is a renowned technique widely used for control of non-linear systems [44, 75]. This technique is based on the Lyapunov theory. Its control objective is to force the convergence of a set of predefined errors to zero. For this purpose, a Lyapunov function is stated for each error in such a way that if the time derivative of those functions are negative definite, the stability of the system, and therefore the convergence of the error to zero, is assured. Then, the control actions of the system are obtained as the ones that make negative definite the time derivative of all the stated Lyapunov functions.

This control strategy is used in most of the trajectory tracking control literature, as stated in [68]. Good performance in the reference tracking is achieved, as mentioned in [66], since *Backstepping* is able to provide larger regions of attraction than other types of controllers. However, as quadrotors present an underactuated nature, the control laws that have been developed do not assure global tracking. The solution to this problem is to eliminate the time dependence of the reference path and thus transform the trajectory tracking problem into a path following problem, in which it is possible to obtain a globally convergent *Backstepping* controller while keeping a large control capability [16].

In several publications *Backstepping* technique was used to solve both path following and stabilization problems. See [15] where a *BS* technique is applied using the IGC structure for the 3D control of a quadrotor. The proposed solution consists of a non-linear state feedback controller for thrust and torque control actions and a timing law that maintains the PF control law well-defined. This controller guarantees global asymptotic convergence of the path following error to zero for a wide class of desired paths and ensures that the actuation does not grow unbounded. The performance of the proposed controller is validated with simulation results. In [16] the authors improve their controller to deal with the presence of constant wind disturbance. This controller includes an estimate of the external disturbance to mitigate its effect. Experimental results demonstrate the robustness of the controller.

In [43] the problem of time cooperative PF for multirotors with a suspended payload is addressed. A team of multirotors transporting a suspended payload. A robust PF controller is developed for each vehicle in the system formed by an autopilot that controls the attitude and a *BS* controller that ensures each vehicle follows the desired path along a given speed profile. Numerical results verify path following accuracy and low coordination errors.

3.2 Lyapunov-Based

Similarly to *Backstepping*, these algorithms are based on the Lyapunov theory. They are developed by assuring the Lyapunov stability condition, and thus, the convergence of the controller. The generated guidance control laws are usually simple and effective [20].

In [25] the authors propose a 3D path following control law based on Lyapunov theory by using the rotation matrix, that belongs to the 3D Special Orthogonal group (SO(3)), for attitude representation. This results in a singularity-free solution and allows speed profile's independent adjustment. The control law generates angular rates and thrust reference commands. Experimental results

for the mission of following a desired path and for the time coordination problem [26] are given to illustrate the efficacy of the proposed control law.

In [57] a Lyapunov-based PF controller is proposed and it is complemented with a velocity observer and a constant disturbance estimator based on the immersion and invariance technique. Experimental indoor results for 3D paths show that the proposed approach fulfils the geometric specifications with an error that, according to the authors, is acceptable.

Some of the Lyapunov-based path following algorithms perform well against wind disturbances. As in [58], where an adaptive non-linear path following method is applied on a fixed-wing experimental platform. In this paper, the Lyapunov function is constructed based on the error equations and desired path function applying the vector field theory. Experimental results show good performance under wind disturbances. [19] presents another path following application for a fixed-wing vehicle that also takes into account wind disturbances by including an *Active Disturbance Rejection Control* (ADRC) for the attitude inner-loop combined with the *Lyapunov-based* control for the outer-loop. Experimental flight tests verify the effectiveness of this method.

3.3 Feedback Linearisation

Feedback Linearisation, along with *Backstepping*, is one of the most commonly used techniques for the control of quadrotors as discussed in [4]. The aim of this control technique is to linearise a system in a certain region of the state space by applying a non-linear inversion of the plant, so that non-linearities in the plant are cancelled and linear control theory can be applied [14]. Some of the advantages of this method are the simplicity in the control structure, the facility of implementation [74] and formal profs of error convergence when appropriate conditions are met. When applied to the PF problem, this method achieves the property of path invariance. That is, to ensure that once the system reaches the path it will stay on it for all future time [3, 4].

In [68] the 3D path following problem for a quadrotor is solved by applying input dynamic extension and input-output *Feedback Linearisation* (FL). The designed controller allows to specify the speed on the path and the yaw angle of the vehicle as a function of the displacement along the path. Simulation results for a constant cruise speed along a circular path show that the quadrotor converges to the path as well as the velocity and yaw angle converge to the desired values. In [3] the authors propose an improvement of the previous stated controller by implementing a transverse *Feedback Linearisation* plus input dynamic extension. This enhanced controller, which fully linearises the system, allows the quadrotor to move along the path in any desired direction, and both closed and non-closed curves can be used for the path definition. The capabilities of the controller are demonstrated with simulation results.

In [4], the authors of [3] adapted its PF controller to operate in the context of fault tolerant control, in the particular case when one of the four motors is completely disabled due to a failure. With the faulty system, only partial *FL* can be achieved. That is because the three rotor system is not differentially flat and presents uncontrolled internal dynamics. However, the uncontrolled non-linear dynamics are proved to be bounded, and thus it is shown that the system can be made to stay precisely on the path while the quadrotor is running only on three rotors.

In [32], a 3D PF implementation of a helicopter UAV is described. They apply the Feedback Linearisation technique basing it on the kinematic model of a helicopter. Six *PID* controllers are employed to control the attitude angles and the velocities. The main contribution of such approach is the consideration of the desired speed of the vehicle as a function of the assigned path. Simulations compare the designed controllers with variable and fixed velocity profile. Better performance is achieved whith variable speed profile.

3.4 Geometric

Geometric techniques were initially described in the missile guidance and control literature. Some of them were adapted to other type of vehicles, such as UGVs or UAVs. Examples of these techniques are: *Carrot-Chasing* algorithm, *Non-Linear Guidance Law*, *Pure Pursuit*, *Line-of-sight* and *Trajectory Shaping* guidance law. *Carrot-Chasing* (*CC*) [59] is a simple geometric strategy that consists on steering the UAV toward a Virtual Target Point (VTP) located on the path. The VTP is periodically updated and obtained by adding a constant distance along the path to the vehicle's closest point. *Non-Linear Guidance Law* (*NLGL*) [65] is also based on the VTP concept. In this case the point is calculated by creating a circumference around the vehicle with a constant radius and taking the point in which the circumference intersects the path. The obtained VTP is at a distance from the vehicle equal to the radius of the circumference. Next, the acceleration commands that steer the vehicle to the VTP are calculated. The *Pure Pursuit* (*PP*) [9, 63] algorithm tries to guide the vehicle straight to a target point on the path. *Line-of-sight* (*LOS*) [8, 71] seeks to steer the vehicle directly towards the closest point on the path. *Pure Pursuit and Line-of-sight* (*PLOS*) [45] is the combination of *PP* and *LOS*. In the *Trajectory Shaping* (TS) [67] guidance law, the commanded lateral acceleration is generated as function of the vehicle heading angle, target heading angle and line-of-sight angle.

Two algorithms based on missile guidance laws, *Pure Pursuit* and *Trajectory Shaping*, have been implemented on a quadrotor vehicle in [53]. The concept of VTP navigation has been utilized to generate the required curved trajectories. A guidance algorithm based on the notion of Differential Flatness (DF) was implemented as a baseline control-oriented algorithm. Simulation and experimental results comparing these algorithms show that *TS* generates smaller position errors and requires lower control effort than *PP* and *DF* approaches. Furthermore, it is proved that missile guidance laws can be successfully utilized by a quadrotor.

In [34] various geometric algorithms, such as *PP*, *LOS* and *Proportional Navigation Guidance* (PPN) law, are applied to the problem of autonomous landing of a quadrotor UAV. Simulation results suggest that the *PPN* algorithm performs the best in terms of time and control effort, where *LOS* presents the worst performance.

In [78] a survey of some of the most common PF algorithms applied to fixed-wing UAVs is presented. In addition to the control-oriented algorithms, *Carrot-Chasing*, *NLGL* and *PLOS* are described. Simulation results comparing those algorithms show that the *NLGL* is the geometric approach that achieves the best performance in terms of path distance error.

3.5 Model Predictive Control

Model Predictive Control (MPC) is a well-known technique [17, 33, 56] that transforms the control problem into an optimization problem. At any sampling time instant, a sequence of future control values is computed by solving a finite horizon optimal control problem. Only the first element of the computed control sequence is used and the overall process is repeated at the next sampling time. The most important drawback of this technique is that resources needed for computation and memory growth rapidly with the time horizon.

In contrast to the most common path following approaches, such as the geometric algorithms or the *Back-stepping* technique, the *MPC* approach is able to handle constraints on states and inputs, non-linear MIMO dynamics, and non-linear reference paths [31].

In [62] a cascaded control structure applying the *Non-linear MPC* technique for the PF outer-loop control is presented. The inner-loop control for the acceleration tracking is based on non-linear dynamic inversion. This controller allows multirotor vehicles to follow a path whose geometry is defined as a spline in 3D space. Simulation results demonstrate the good performance of this approach. Furthermore, the obtained runtimes, well below the sample rate, suggest that it could be implemented in on-board embedded systems. An improvement of this approach is presented in [5], where an adaptive augmentation scheme for the inner-loop controller is designed. The adaptive augmentation is based on a non-linear design plant and uses *Model Reference Adaptive Control* (MRAC). Simulation results show a stronger robustness of the adaptive augmentation in relation to the baseline controller.

A trajectory optimization strategy for UAVs based on non-linear optimal control techniques is presented in [69]. The approach is based on a Virtual Targed Vehicle (VTV) perspective where a virtual target is introduced. Numerical computation results show that it is able to compute aggressive manoeuvres. In [70] the authors extend and adapt the proposed strategy for path following in presence of time varying wind disturbances by means of a sample-data *MPC* architecture. Simulation results with a fixed-wing vehicle show the effectiveness of the predictive PF approach.

In [37] a non-linear receding horizon guidance law is developed to solve the PF problem on a fixed-wing vehicle. An extended Kalman filter is used to estimate wind velocities. The proposed law achieves efficient PF by making input constraints active. Its effectiveness is demonstrated by flight tests.

In addition to path following approaches, several trajectory tracking implementations on quadrotor vehicles are found applying the *Model Predictive Control* technique. Some of them have reported experimental results showing a good performance against disturbances [6, 11].

3.6 Vector Field

In the *Vector Field* (*VF*) based control, a set of vectors is virtually placed around the path in such a way that if the vehicle follows the direction of those vectors it will converge into the path. Those vectors are used to generate desired course inputs to the inner-loop attitude controllers.

The first application of a path following approach using this method for a UAV is found in [60]. In this paper *Vector Field* path following control laws are developed for straight-line paths and circular arcs and orbits. Asymptotic decay of path following errors in the presence of constant wind disturbances is demonstrated with Lyapunov theory. Experimental tests are carried out with a fixed-wing UAV to show the effectiveness of this method.

In [88] the 3D path following problem is solved by implementing a velocity vector field following controller based on the differential flatness notion. This approach is designed with an inner-loop controller that makes the vehicle follow a specified velocity vector field. The validity of the method is demonstrated by numerical simulations and experiments with a quadrotor for three different vector fields.

A *VF* path following guidance for 2D and 3D twice differentiable curves is stated in [49]. The proposed

approach is based on the Helmholtz theorem, which states that an arbitrary vector field can be decomposed into two parts, conservative part (irrotational) and solenoidal part (rotational). This approach combines both parts by using the conservative part for long distances to the desired path and using the solenoidal part when the vehicle is along the path. UAV input constraints and constant wind disturbances are assumed to be present. The method's performance is validated by simulation results.

In [87] an adaptive control scheme for UAVs path following under wind disturbances is proposed. This control strategy integrates the *Vector Field* PF law with an adaptive term to deal with the effect of unknown wind disturbances. Simulation results with wind conditions show that the proposed method compensates for the lack of knowledge of the wind vector and, according to the authors, it attains a smaller path following error than the state-of-art vector field method.

Refer to [35, 41, 54, 82, 83] for path following *Vector Field* implementations applied on different types of aerial vehicles. In [78], simulation results comparing this algorithm to other geometric and control-oriented algorithms show that the *VF-based* algorithm is the one with least cross-track error. However, final remarks conclude that it can be difficult to implement.

3.7 Learning-Based

Learning-based algorithms constitute an emerging field that, due to the significant progress made in recent years, has become a wise solution for different types of problems. In particular, it is being introduced in the trajectory control problem on different types of vehicles, including UAVs. This field includes supervised and unsupervised techniques, reinforcement learning algorithms, data-driven approaches and other deep machine learning methods.

In [76] a *MPC* controller is used as a supervisor to train a neural network control policy to solve the path following and obstacle avoidance problem on a quadrotor vehicle. With this algorithm the computational efficiency problem of the *MPC* technique is solved while maintaining a similar performance, as proved by experimental results. An Iterative Learning Control approach is proposed in [86] to solve the PF problem on a quadrotor. The control scheme is composed by a PD controller and an anticipant controller. This approach focuses on repetitive flight to learn from experience. Simulation tests and off-line experimental results are presented to prove the effectiveness of the controller. In [85] a data-driven control approach is proposed for the adaptive path following control of a fixed-wing vehicle. The reliability of the approach is demonstrated through simulation results and flight tests. Other learning-based approaches have been used to solve the path following problem on different types of vehicles, such as vessels [36, 55, 73] or airships [61].

Various learning-based approaches are found in the literature solving the trajectory tracking problem for a quadrotor vehicle [24, 79]. Some of these approaches consider wind disturbances in its design, as in [51] where an adaptive trajectory tracking control based on a reinforcement learning algorithm is presented.

3.8 Optimal Control

The Optimal Control theory aims to operate a dynamic system at a minimum cost. That is, to follow a path with a minimum error and control effort. The most well-known control techniques to solve this problem are the *Linear Quadratic Regulator* (LQR) and the *Linear Quadratic Gaussian* (LQG).

In [80] a general solution for the PF problem is presented. The proposed approach is developed as a fixed end-time optimal control problem and it relies on a geometric formulation based on the notion of differential flatness. The resulting controller is applied to a quadrotor simulation system with the mission of performing aggressive manoeuvres and it is demonstrated that the proposed problem formulation is solved efficiently.

A UAV guidance law using an adaptive *LQR* formulation is addressed in [46]. The *LQR* is optimized using a genetic algorithm for tighter control of UAV errors in high disturbances. Simulations for straight line and loiter paths under various wind conditions prove the effectiveness of the approach.

Experimental results applying optimal control theory to solve the trajectory tracking problem on a quadrotor vehicle can be found in [39]. In this paper, the authors propose a solution based on the definition of a path-dependent error space to express the dynamic model of the vehicle. The controller is designed using *LQR* state space feedback and adopts the *D-methodology* integrated with the anti-windup technique in order to achieve zero static error for the integral states and avoid actuator saturation.

3.9 Sliding Mode Control

Sliding Mode Control (SMC) is a non-linear control method that attains the control objectives by constraining the system dynamics to a pre-defined surface by means of a discontinuous control law. This control technique is considered to be effective and robust [29]. However, it can present implementation issues due to the chattering effect.

The bibliographic search revealed no *SMC* application solving the path following problem for a quadrotor vehicle. However, it has been applied to solve the PF problem

on other UAVs, such as fixed-wing vehicles. In [72] a lateral guidance law for cross-track control based on the *SMC* technique is developed. This guidance law includes a feed-forward component related to the rate of change of the desired heading angle, which permits to improve the performance and achieve accurate tracking while following curved paths. The proposed guidance scheme is evaluated based on experimental flight results. According to the authors, it presents a good performance in the presence of wind and parametric uncertainty. In [7] the controller presented in [72] is improved by implementing a Partially-IGC strategy that includes a sliding mode control on the inner and the outer loops using a non-linear sliding surface based on *Second Order Sliding Mode* (*SOSM*) control theory. Experimental tests compare the conventional SGC approach and the proposed Partially-IGC approach to show that the second one presents a faster convergence of the cross-track error toward zero.

In [84] the Second Order Sliding structure is used to develop a PF application. The proposed controller provides smooth bank and turn coupled motions. To estimate the uncertain sliding surfaces a *High-Order Sliding Mode* (*HOSM*) differentiator is applied. The sliding surface structure is based on the Pure Pursuit algorithm through a set of intermediate control variables and also introducing a virtual target point in the path. This approach eliminates time-consuming and intensive computation. According to the authors, simulations show that it provides an excellent performance even under wind turbulence conditions.

Regarding the trajectory tracking problem, numerous *SMC* quadrotor implementations are found [12, 85]. Some of these approaches are able to deal with wind disturbances [30, 81].

3.10 Comparison

Refer to Table 1 for a comparison of the reviewed PF algorithms. The characteristics of these algorithms are evaluated only in the context of the path following problem applied to UAVs and are based on the reviewed literature. The columns refer respectively to: the control structure (i.e. Integrated Guidance and Control or Separated Guidance and control); the type of results (experimental or simulation); the application to quadrotors; good experimental results against external disturbances; including a Fault Tolerant Control (FTC) strategy; and implementing an adaptive approach.

4 Implementation of Path Following Algorithms on a Quadrotor

In this section, four path following algorithms have been chosen for implementation on the quadrotor vehicle: *Backstepping* and *Feedback Linearization*, which are the two most referenced control algorithms, followed by *Non-Linear Guidance Law* (*NLGL*) and *Carrot-Chasing* geometric algorithms.

In the reviewed literature, no application to quadrotors has been reported of the *NLGL* and *Carrot-Chasing* algorithms. However, they are commonly used on fixed-wing vehicles [65, 78], as well as on Unmanned Ground Vehicles (UGV) [59, 63], and sometimes on other types of unmanned vehicles. They are simple and they typically provide feasible solutions with good path following performance. Both these geometric algorithms are very similar, the only difference is in how the VTP is calculated. In this paper such simple and fruitful algorithms are implemented to evaluate their performance.

Our own implementation of these algorithms will provide an evaluation and comparison under the same conditions. Refer to Section 5, where the simulation results are presented.

4.1 Dynamic Model of the Quadrotor

In this section, the mathematical model of the quadrotor is presented. It is a standard non-linear model, which can be found explained in more detail in the literature [13, 52].

Table 1 Comparison of the PF techniques

	Structure	Results	Quadrotor	Wind dist.	FTC	Adaptive
Backstepping	IGC	Experimental	Yes	Yes	No	No
Lyapunov-based	SGC and IGC	Experimental	Yes	Yes	No	Yes
Feedback Linearization	SGC and IGC	Simulation	Yes	No	Yes	No
Geometric	SGC	Experimental	Yes	No	No	No
Model predictive control	SGC	Experimental	Yes	Yes	No	Yes
Vector field	SGC	Experimental	Yes	Yes	No	Yes
Learning-based	SGC	Experimental	Yes	Yes	No	Yes
Optimal control	SGC and IGC	Simulation	No	Yes	No	Yes
Sliding mode control	SGC and IGC	Experimental	No	Yes	No	No

Notation In this paper bold lower-case letters indicate vectors and bold upper-case letters indicate matrices. Same is applied for vector and matrix functions. The top right superscript, occasionally employed, indicates the variable's frame of reference.

The coordinate systems as well as the axis labels and rotational conventions defined in this dynamic model are shown in Fig. 2. Two coordinate systems can be found: The body frame of reference $\{B\}$, which is attached to the body of the quadrotor, and the world reference frame $\{W\}$, considered inertial.

This model has four inputs and twelve states. The inputs are the total thrust and the torques applied on each rotational axis (T, τ_ϕ, τ_θ, τ_ψ). The states are the body velocities (u, v and w), the world position (x, y and z), the body angular velocities (p, q and r) and the Euler angles (ϕ-*roll*, θ-*pitch* and ψ-*yaw*).

$$\dot{\mathbf{v}} = \frac{1}{m}\left(\mathbf{F} - \mathbf{F_d} - \mathbf{F_w}\right) + \mathbf{g}^B - \boldsymbol{\omega} \times \mathbf{v} = \begin{bmatrix} \dot{u} & \dot{v} & \dot{w} \end{bmatrix}^T \tag{1}$$

$$\dot{\mathbf{p}} = \mathbf{R}\mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{2}$$

$$\dot{\boldsymbol{\omega}} = (\mathbf{J})^{-1}\left[\mathbf{M} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}\right] = \begin{bmatrix} \dot{p} & \dot{q} & \dot{r} \end{bmatrix}^T \tag{3}$$

$$\dot{\boldsymbol{\Phi}} = \mathbf{H}\,\boldsymbol{\omega} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{4}$$

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & c_T\left(\omega_{m1}^2 + \omega_{m2}^2 + \omega_{m3}^2 + \omega_{m4}^2\right) \end{bmatrix}^T \tag{5}$$

$$\mathbf{M} = \begin{bmatrix} d\,c_T\,\omega_{m2}^2 - d\,c_T\,\omega_{m4}^2 + J_m\,q\,(\pi/30)\,(\omega_{m1} - \omega_{m2} + \omega_{m3} - \omega_{m4}) \\ -d\,c_T\,\omega_{m1}^2 + d\,c_T\,\omega_{m3}^2 + J_m\,p\,(\pi/30)\,(-\omega_{m1} + \omega_{m2} - \omega_{m3} + \omega_{m4}) \\ -c_Q\,\omega_{m1}^2 + c_Q\,\omega_{m2}^2 - c_Q\,\omega_{m3}^2 + c_Q\,\omega_{m4}^2 \end{bmatrix} \tag{6}$$

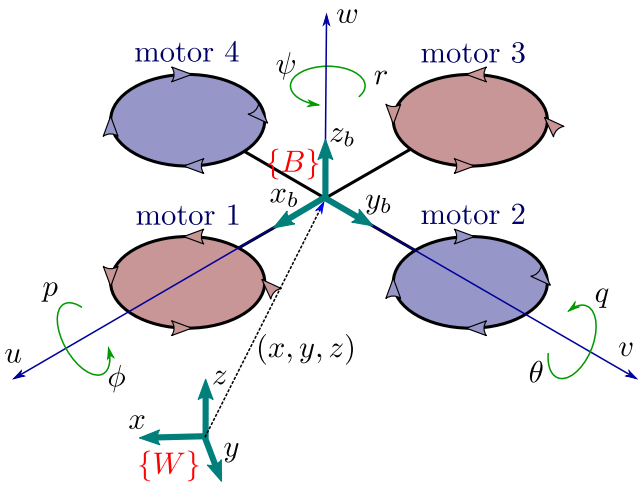

Fig. 2 Axis labels and conventions

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & d\,c_T & 0 & -d\,c_T \\ -d\,c_T & 0 & d\,c_T & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} \omega_{m1}^2 \\ \omega_{m2}^2 \\ \omega_{m3}^2 \\ \omega_{m4}^2 \end{bmatrix} \tag{7}$$

Equations 1–6 define the dynamic model of the quadrotor. A brief description of the parameters of these equations is given in Table 2. Equation 1 is the linear velocity state equation. They are the linear velocities that correspond to the x, y and z axis of $\{B\}$, respectively.

The position state vector is calculated in Eq. 2. The term $\mathbf{R}$ is a rotation matrix to transform from the body to the world frame using the rotational sequence zyx.

Equation 3 defines the angular velocity state update. p, q and r are the rate change of *roll*, *pitch* and *yaw*, respectively, represented in the body frame. Equation 4 is the Euler kinematic equation that determines the rate of change of the Euler angles in the world frame. Matrix $\mathbf{H}$ relates the angular velocity of the aircraft in the body frame with the changes in angle and it is obtained by using sequential rotation matrices (*pitch-roll-yaw* sequence).

Equation 5 shows the thrust forces, always parallel to the z body axis, where ω_{mi} is the rotational velocity of motor i. Equation 6 defines the generated gyroscopic and thrust moments on each body axis.

Equation 7 presents the relation between the rotation speed of the motors (ω_{m1-4}) and the total thrust, T, and the moments τ_ϕ, τ_θ, and τ_ψ applied on each rotational axis (*roll*, *pitch* and *yaw*).

4.2 Backstepping

The *Backstepping* controller developed in this paper is an adaptation of the one used in [15]. Note that the UAV model used in [15] is slightly different from the one defined in Section 4.1. The frames of reference considered in each model are different and [15] uses a rotation matrix to represent the attitude.

As in [15], the gyroscopic effects have been omitted to perform the calculations to obtain the control law. However, to validate the algorithm properly, these effects are taken into account in the simulator model.

Table 2 Variables of the model

Symbol	Description
m	Mass of the quadrotor.
$\mathbf{g}^B$	Acceleration of gravity expressed in the body frame.
$\mathbf{J}$	Mass moment of inertia matrix.
c_T	Thrust coefficient.
c_Q	Torque coefficient.
d	Distance from a given motor to the center of gravity.
J_m	Inertia of each motor's rotating components.

In this algorithm, $\mathbf{S}()$ represents the skew symmetric matrix that verifies $\mathbf{S}(\mathbf{x})\mathbf{y} = \mathbf{x} \times \mathbf{y}$. And $\mathbf{p}'_d$ is the partial derivative of $\mathbf{p}_d$ with respect to γ, $\frac{\partial \mathbf{p}_d}{\partial \gamma}$.

To solve the path following problem, four backstepping error vectors are defined, each one of three components. The first error vector, Eq. 8, is the position error. The second error vector, Eq. 9, includes a term of position error and a term of velocity error in the world frame. The sigmoidal function $\sigma(\cdot)$, Eq. 10, limits the growth of $\mathbf{e}_2$ when there are large position errors, where $p_{\max}$ is the allowed limit. This guarantees that the actuation does not grow unbounded.

$$\mathbf{e}_1 = \mathbf{p} - \mathbf{p}_d(\gamma) \tag{8}$$

$$\mathbf{e}_2 = \boldsymbol{\sigma}(\mathbf{e}_1) + \frac{1}{k_1}\dot{\mathbf{e}}_1 \tag{9}$$

$$\sigma(\mathbf{x}) = p_{\max}\frac{\mathbf{x}}{1 + \|\mathbf{x}\|} \tag{10}$$

The convergence of $\mathbf{e}_1$ and $\mathbf{e}_2$ to zero is assured by defining two control Lyapunov functions, whose derivatives are negative definite if the thrust force, T, follows T_d, Eq. 11, with the direction $\mathbf{r}_{3d}$, Eq. 12, where $\mathbf{u}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

$$T_d = m\left\| k_1^2 k_2 \mathbf{e}_2 + k_1 \dot{\boldsymbol{\sigma}}(\mathbf{e}_1) + g\mathbf{u}_3 - \ddot{\mathbf{p}}_d \right\| \tag{11}$$

$$\mathbf{r}_{3d} = \frac{k_1^2 k_2 \mathbf{e}_2 + k_1 \dot{\boldsymbol{\sigma}}(\mathbf{e}_1) + g\mathbf{u}_3 - \ddot{\mathbf{p}}_d}{\left\| k_1^2 k_2 \mathbf{e}_2 + k_1 \dot{\boldsymbol{\sigma}}(\mathbf{e}_1) + g\mathbf{u}_3 - \ddot{\mathbf{p}}_d \right\|} \tag{12}$$

The control law expression for the thrust force is calculated as shown in Eq. 13. $\mathbf{r}_3$ is the third column of $\mathbf{R}$, which represents the direction of the z body axis of the vehicle. Thus, if $\mathbf{r}_3$ is equal to the desired thrust direction $\mathbf{r}_{3d}$, the thrust force will be the same as the desired thrust force T_d. For this reason, the third error vector is defined as the error of the thrust direction, as stated in Eq. 14.

$$T = \mathbf{r}_{3d}^T \mathbf{r}_3 T_d \tag{13}$$

$$\mathbf{e}_3 = \mathbf{r}_3 - \mathbf{r}_{3d} \tag{14}$$

A third control Lyapunov function, which includes the first and second Lyapunov functions as well as a term for the third backstepping error, is defined. With the aim of making the first three errors tend to zero, the expression of the fourth error vector, Eq. 15, is defined in such a way that if this error becomes zero, the time derivative of the third Lyapunov function remains negative definite. In Eq. 15 $\mathbf{R}_d$ represents the desired orientation matrix and $\boldsymbol{\omega}_d$ stands for the desired angular speed of the vehicle.

$$\begin{aligned}\mathbf{e}_4 = {} & -k_3 \mathbf{S}(\mathbf{u}_3)^2 \mathbf{R}^T \mathbf{r}_{3d} + \mathbf{S}(\mathbf{u}_3)(\boldsymbol{\omega} - \mathbf{R}^T \mathbf{R}_d \boldsymbol{\omega}_d) \\ & - \tfrac{T_d}{mk_1}\mathbf{S}(\mathbf{u}_3)^2 \mathbf{R}^T \mathbf{e}_2\end{aligned} \tag{15}$$

Finally, a fourth Lyapunov function is defined for the fourth error. To assure the stability of the system with this Lyapunov function, the expression of the angular acceleration of the vehicle is calculated as stated in Eq. 16. In this expression $\dot{\boldsymbol{\omega}}_c$ stands for the calculated angular acceleration and $\dot{\omega}_{3c}(t)$ is an arbitrary function that defines the dynamics of the *yaw* angle. Then, the control law of the torque can be calculated from this expression by means of Eq. 17.

$$\begin{aligned}\dot{\boldsymbol{\omega}}_c = {} & -\mathbf{S}(\mathbf{u}_3)\Big(- k_4 \mathbf{e}_4 - \mathbf{R}^T \mathbf{r}_{3d} + k_3 \mathbf{S}(\mathbf{u}_3)^2 \big(\dot{\mathbf{R}}^T \mathbf{r}_{3d} + \mathbf{R}^T \dot{\mathbf{r}}_{3d}\big) \\ & + \tfrac{1}{mk_1}\mathbf{S}(\mathbf{u}_3)^2 \big(\dot{T}_d \mathbf{R}^T \mathbf{e}_2 + T_d \dot{\mathbf{R}}^T \mathbf{e}_2 + T_d \mathbf{R}^T \dot{\mathbf{e}}_2\big)\Big) \\ & + \dot{\mathbf{R}}^T \mathbf{R}_d \boldsymbol{\omega}_d + \mathbf{R}^T \mathbf{R}_d \dot{\boldsymbol{\omega}}_d + \begin{bmatrix} 0 & 0 & \dot{\omega}_{3c}(t) \end{bmatrix}^T\end{aligned} \tag{16}$$

$$\boldsymbol{\tau} = \mathbf{J}\dot{\boldsymbol{\omega}}_c + \mathbf{S}(\boldsymbol{\omega})\mathbf{J}\boldsymbol{\omega} \tag{17}$$

It can be noticed that constants k_{1-4} appear in the calculated control laws. These constants define the dynamics of each backstepping error.

More details about the derivation of these control laws and about their stability proofs are given in [15].

Notice that defining the error vectors this way sets only the third column of $\mathbf{R}_d$. This is clear by checking the definitions of error $\mathbf{e}_3$ (in Eq. 14) or function $\dot{\omega}_{3c}(t)$, that appears in the expression of the calculated angular velocity (16). Thus, the direction of the x and y axis can be assigned arbitrarily, as long as the orthogonality of the frame of reference is satisfied, i.e. a degree of freedom appears for the evolution of *yaw*. Exploiting this degree of freedom, the desired rotation matrix $\mathbf{R}_d$ has been defined as the one that makes the velocity vector of the vehicle tangent to the path, as shown in Eq. 18. To assure this control specification, a PD-like controller is designed as a control law for function $\dot{\omega}_{3c}$, as shown in Eq. 19, where $\mathbf{r}_1$ is the first column of the rotation matrix $\mathbf{R}$ and $\mathbf{r}_{2d}$ is the second column of the desired rotation matrix $\mathbf{R}_d$.

$$\mathbf{R}_d = \begin{bmatrix} -\frac{S(\mathbf{r}_{3d})^2 \mathbf{p}'_d}{\|S(\mathbf{r}_{3d})^2 \mathbf{p}'_d\|} & \frac{S(\mathbf{r}_{3d})\mathbf{p}'_d}{\|S(\mathbf{r}_{3d})\mathbf{p}'_d\|} & \mathbf{r}_{3d} \end{bmatrix} \tag{18}$$

$$\dot{\omega}_{3c} = -l_2\left(\omega_3 - \omega_{3d} + l_1 \mathbf{r}_{2d}^T \mathbf{r}_1\right) + \dot{\omega}_{3d} - l_1 \tfrac{d}{dt}\left(\mathbf{r}_{2d}^T \mathbf{r}_1\right) \tag{19}$$

Parameter γ (virtual arc parameter, see Definition 1) is present in the definition of $\mathbf{e}_1$, Eq. 8, and implicit in the rest of the controller expressions. This parameter defines the desired trajectory position, velocity and acceleration at a certain time instant. Therefore, the controller algorithm needs to have this parameter properly scheduled to work correctly. This is known as the timing law. The timing law stated in this algorithm is given by the second time derivative of γ, defined in Eq. 20. Matrix ${}^W_T\mathbf{R}^T(\gamma)$ represents the rotation matrix from a frame $\{T\}$ tangent to the path to the world frame, obtained as shown in Eq. 21, where $\xi(\gamma)$ is a function that changes sign in the inflection points of the path. This timing law has been defined to make the time derivative of γ converge to its desired value $\dot{\gamma}_d$, and also to

preserve the stability of the system as long as the condition stated in Eq. 22 is satisfied. That condition assures the ascending behaviour of the path. $\dot{\gamma}_d$ is a control specification that can be related to the path's shape. Examples of the definition of this variable are given in Section 5.

$$\ddot{\gamma} = -k_\gamma \sigma(\dot{\gamma} - \dot{\gamma}_d) + \mathbf{u}_1^T {}_T^W\mathbf{R}^T \left(k_1^2 k_2 \mathbf{e}_2 + k_1 \dot{\boldsymbol{\sigma}}(\mathbf{e}_1) - \dot{\gamma}\mathbf{p}_d''\right) \tag{20}$$

$${}_T^W\mathbf{R}^T(\gamma) = \left[\frac{\mathbf{p}_d'(\gamma)}{\|\mathbf{p}_d'(\gamma)\|} \; \xi(\gamma) \frac{S(\mathbf{p}_d'(\gamma))^2 \mathbf{p}_d''(\gamma)}{\|S(\mathbf{p}_d'(\gamma))^2 \mathbf{p}_d''(\gamma)\|} \; \xi(\gamma) \frac{S(\mathbf{p}_d'(\gamma))\mathbf{p}_d''(\gamma)}{\|S(\mathbf{p}_d'(\gamma))\mathbf{p}_d''(\gamma)\|} \right] \tag{21}$$

$$\left|\mathbf{p}_d'(\gamma)^T \mathbf{u}_3\right| > \alpha \tag{22}$$

Some of the mathematical expressions stated for the *Backstepping* algorithm require derivatives that have to be estimated in order to avoid their arduous analytical calculations. In this paper, a first order filter was used to obtain a time derivative estimation of a known variable. The scheme of this filter is shown in Fig. 3, where x is the known variable, $\hat{x}$ is the filtered variable, $\dot{\hat{x}}$ is the time derivative estimation of x and x_0 is its initial value.

The filter was used to estimate the time derivative of the thrust ($\dot{T}_d$) and the three components of the direction vector of this force ($\dot{\mathbf{r}}_{3d}$). With $\dot{\mathbf{r}}_{3d}$ it is possible to calculate $\dot{\mathbf{R}}_d$ as shown in Eq. 23. The angular speed vector $\boldsymbol{\omega}_d$ can be obtained from its skew symmetric matrix, calculated as shown in Eq. 24. The time derivative of the desired angular speed $\dot{\boldsymbol{\omega}}_d$ can be estimated using an estimation filter for each component of vector $\boldsymbol{\omega}_d$.

$$\dot{\mathbf{R}}_d = \frac{\partial \mathbf{R}_d}{\partial \mathbf{r}_{3d}} \dot{\mathbf{r}}_{3d} + \frac{\partial \mathbf{R}_d}{\partial \mathbf{p}_d'} \mathbf{p}_d'' \dot{\gamma} \tag{23}$$

$$\mathbf{S}(\boldsymbol{\omega}_d) = \mathbf{R}_d^T \dot{\mathbf{R}}_d \tag{24}$$

The control structure of the *BS* controller is shown in Fig. 4. It is an IGC structure, as it is implemented without any inner loop controller. The second derivative of γ, calculated by the timing law, is integrated twice to obtain $\dot{\gamma}$ and γ. The derivative estimation filters have been omitted to clarify the scheme.

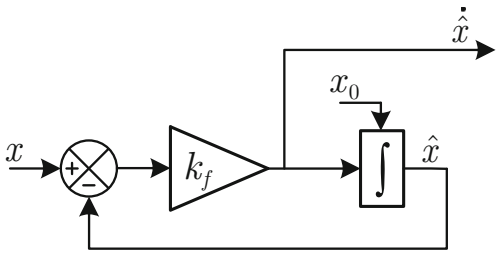

Fig. 3 General derivative estimation diagram

The implementation of the *Backstepping* controller is given in Algorithm 1. This algorithm requires the state vector of the system ($\mathbf{x}$), the estimated derivative parameters ($\hat{\dot{T}}_d, \hat{\dot{r}}_{3d}, \hat{\dot{\omega}}_d$), the values of γ, its derivative $\dot{\gamma}$ and its desired derivative $\dot{\gamma}_d$, the control design parameters (k_{1-4}, $p_{\max}$, l_{1-2}, k_γ), the desired path ($\mathbf{p}_d(\gamma)$) and the first and second partial derivatives of the desired path with respect to γ ($\mathbf{p}_d'(\gamma), \mathbf{p}_d''(\gamma)$). The algorithm returns the control signals of thrust and torques (T, τ_ϕ, τ_θ and τ_ψ). The first step of this algorithm obtains the position ($\mathbf{p}$), velocity ($\mathbf{v}$) and angular velocity ($\boldsymbol{\omega}$) as well as the rotational matrix ($\mathbf{R}$) from the system's states. Next, the set of equations of the algorithm are applied in a specific sequence aiming to obtain the control actions of the thrust and torque forces.

Notice that the components of the y and z axis of each state are made negative to obtain the vectors and rotational matrix and so are these components of the torque vector once the control actions are obtained. This is equivalent to a 180 rotation of the reference frame. As explained at the beginning of this section, both models consider different frames of reference. In order to use the equations as stated in [15] this rotation is applied to the states before and after applying the algorithm.

4.3 Feedback Linearisation

The *Feedback Linearisation* path following algorithm implemented in this paper is based on the development found in [3], which is an improvement of the one in [68]. The employed mathematical model is very similar to one defined in Section 4.1. Thus, the same control design process can be used. Nevertheless, note that in [3] the velocities of the vehicle are defined with respect to the world frame. Therefore, Eq. 25 is now used to define the evolution of these states, where the W on the superscript indicates that is referenced to the world frame. And thus, the evolution of the world position is equivalent to the world velocities, as shown in Eq. 26. Also, note that in this paper a *zyx*-sequence is used to define the rotation matrix (Section 4.1), while [3] uses a *zxy*-sequence. However, the selection of this matrix does not affect the behaviour of the system as long as the matrix is well-defined.

$$\dot{\mathbf{v}}^W = \left(\frac{1}{m}\right) \mathbf{R}\,\mathbf{F} - \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T = \begin{bmatrix} \dot{u}^W & \dot{v}^W & \dot{w}^W \end{bmatrix}^T \tag{25}$$

$$\dot{\mathbf{p}} = \mathbf{v}^W = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T \tag{26}$$

As in the *Backstepping* algorithm, the gyroscopic effects have been omitted to obtain the equations of this algorithm.

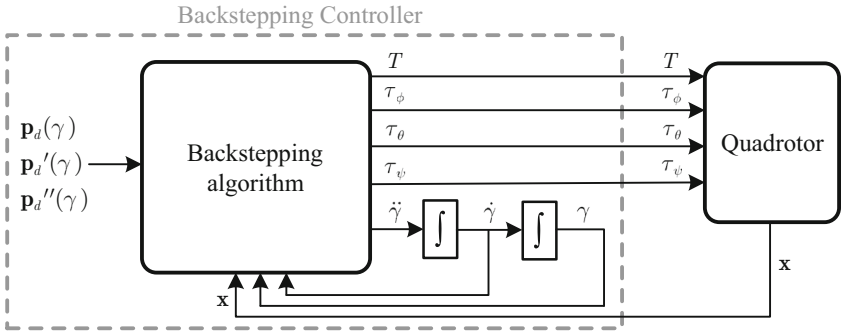

Fig. 4 Backstepping control structure

Equation 27 defines the system that will be used to apply the *FL* control.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{u}^W \\ \dot{v}^W \\ \dot{w}^W \\ \dot{p} \\ \dot{q} \\ \dot{r} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} u^W \\ v^W \\ w^W \\ 0 \\ 0 \\ -g \\ q\,r\,J_1 \\ p\,r\,J_2 \\ p\,q\,J_3 \\ p + \tan(\theta)\,R_4 \\ q\,\cos(\phi) + r\,\sin(\phi) \\ \sec(\theta)\,R_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ R_1 & 0 & 0 & 0 \\ R_2 & 0 & 0 & 0 \\ R_3 & 0 & 0 & 0 \\ 0 & C_x & 0 & 0 \\ 0 & 0 & C_y & 0 \\ 0 & 0 & 0 & C_z \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{27}$$

Where J_1, J_2 and J_3 are inertial terms defined in Eqs. 28-30; R_1, R_2, R_3 and R_4 are auxiliary terms, Eqs. 31–34; and C_x, C_y, C_z are the inverse of the inertia in each coordinate axis, Eq. 35.

$$J_1 = \left(\frac{J_{yy} - J_{zz}}{J_{xx}}\right) \tag{28}$$

$$J_2 = \left(\frac{J_{zz} - J_{xx}}{J_{yy}}\right) \tag{29}$$

$$J_3 = \left(\frac{J_{xx} - J_{yy}}{J_{zz}}\right) \tag{30}$$

$$R_1 = \frac{\sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi)}{m} \tag{31}$$

$$R_2 = \frac{-\sin(\phi)\cos(\psi) + \cos(\phi)\sin(\theta)\sin(\psi)}{m} \tag{32}$$

$$R_3 = \frac{\cos(\phi)\cos(\theta)}{m} \tag{33}$$

$$R_4 = \sin(\phi)\,q + \cos(\phi)\,r \tag{34}$$

$$C_x = \frac{1}{J_{xx}}, \quad C_y = \frac{1}{J_{yy}}, \quad C_z = \frac{1}{J_{zz}} \tag{35}$$

Besides the dynamics of the system, the outputs of interest need to be defined. In this case, a set of four virtual outputs, Eq. 36, are stated. This algorithm depends explicitly on the path projection on the xy plane. According to the simulations planed in Section 5, this projection must be a circumference.

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ h_3(\mathbf{x}) \\ h_4(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} x^2 + y^2 - A^2 \\ z - f_z(\mathbf{x}) \\ A\,\arctan\left(\frac{y}{x}\right) \\ \psi \end{bmatrix} \tag{36}$$

The virtual output h_1 tracks the position error of the vehicle in the xy plane, where A is the radius of the circumference. h_2 tracks the position error in the z coordinate, where $f_z(\mathbf{x})$ is a function that defines the altitude reference. The third virtual output returns a scalar, $h_3 \in [0, 2\pi A]$ in meters, that is the position on the path at a minimum distance in the xy projection. Finally, h_4 is the *yaw* angle.

Defining the outputs this way, the position error can be eliminated by making h_1 and h_2 converge to zero. The point on the path to be tracked by the vehicle can be controlled with h_3, as well as the velocity of this point if the time derivative of h_3 is regulated too. And clearly, the *yaw* angle can be regulated by controlling h_4.

The vector relative degree of the system given by the dynamics in Eq. 27 and the outputs in Eq. 36 is equal to {2, 2, 2, 2}. This vector represents, for each output, the number of time derivatives that need to be done to h to get an explicit relation with one of the inputs. However, as demonstrated in [68] and [3], the vector relative degree is not well-defined. This is because the so called decoupling matrix becomes always singular. That means that the linearised system is not fully controllable. Two new states (ζ_1 and ζ_2) are added to the system to solve this problem. These states satisfy Eq. 37. The total thrust, T, is now a state of the system and the new input u_1 is equivalent to the second derivative of this state.

$$\begin{aligned} &T = \zeta_1, \quad \dot{\zeta}_1 = \zeta_2, \quad \dot{\zeta}_2 = u_1, \\ &\tau_\phi = u_2, \quad \tau_\theta = u_3, \quad \tau_\psi = u_4 \end{aligned} \tag{37}$$

Adding these states to the system, the vector relative degree becomes {4, 4, 4, 2}, which derives in a well-defined decoupling matrix. Since the total number of states of the system is 14 (12 states of the original system and 2 new states) and the vector relative degree adds up to 14,

Algorithm 1 Backstepping.

Require: $\mathbf{x} := \{x, y, z, u, v, w, p, q, r, \phi, \theta, \psi\}$,
$\hat{T}_d$, $\hat{\mathbf{r}}_{3d}$, $\hat{\boldsymbol{\omega}}_d$, γ, $\dot{\gamma}$, $\dot{\gamma}_d$, k_{1-4}, p_{max}, l_{1-2}, k_γ,
$\mathbf{p}_d(\gamma) := [x_d(\gamma),\ y_d(\gamma),\ z_d(\gamma)]^T$, $\mathbf{p}'_d(\gamma)$, $\mathbf{p}''_d(\gamma)$

Create the vectors and rotation matrix from the states:
$\mathbf{p} = [x, -y, -z]^T$, $\mathbf{v} = [u, -v, -w]^T$,
$\boldsymbol{\omega} = [p, -q, -r]^T$, $\mathbf{R}(\phi, -\theta, -\psi)$

Calculate $\dot{\mathbf{p}}_d$
$\dot{\mathbf{p}}_d = \mathbf{p}'_d\dot{\gamma}$

Calculate $\mathbf{e}_1$, $\dot{\mathbf{e}}_1$, $\boldsymbol{\sigma}(\mathbf{e}_1)$, $\dot{\boldsymbol{\sigma}}(\mathbf{e}_1)$ and $\mathbf{e}_2$
$\mathbf{e}_1 = \mathbf{p} - \mathbf{p}_d$, $\dot{\mathbf{e}}_1 = \mathbf{R}\mathbf{v} - \dot{\mathbf{p}}_d$
$\boldsymbol{\sigma}(\mathbf{e}_1) = p_{max}\frac{\mathbf{e}_1}{1+\sqrt{e_{11}^2+e_{12}^2+e_{13}^2}}$
$\dot{\boldsymbol{\sigma}}(\mathbf{e}_1) = \frac{\partial\boldsymbol{\sigma}(\mathbf{e}_1)}{\partial\mathbf{p}}\mathbf{R}\mathbf{v} + \frac{\partial\boldsymbol{\sigma}(\mathbf{e}_1)}{\partial\mathbf{p}_d}\dot{\mathbf{p}}_d$
$\mathbf{e}_2 = \boldsymbol{\sigma}(\mathbf{e}_1) + \frac{1}{k_1}\dot{\mathbf{e}}_1$

Calculate ${}^W_T\mathbf{R}^T$, $\ddot{\gamma}$ and $\ddot{\mathbf{p}}_d$
${}^W_T\mathbf{R}^T$ *from Eq. 21*
$\ddot{\gamma} = -k_\gamma\sigma(\dot{\gamma} - \dot{\gamma}_d) + \mathbf{u}_1^T{}^W_T\mathbf{R}^T\left(k_1^2k_2\mathbf{e}_2 + k_1\dot{\boldsymbol{\sigma}}(\mathbf{e}_1) - \dot{\gamma}\mathbf{p}''_d\right)$
$\ddot{\mathbf{p}}_d = \mathbf{p}''_d\dot{\gamma}^2 + \mathbf{p}'_d\ddot{\gamma}$

Calculate T_d, $\mathbf{r}_{3d}$ and T
$T_d = m\left\|k_1^2k_2\mathbf{e}_2 + k_1\dot{\boldsymbol{\sigma}}(\mathbf{e}_1) + g\mathbf{u}_3 - \ddot{\mathbf{p}}_d\right\|$
$\mathbf{r}_{3d} = \frac{k_1^2k_2\mathbf{e}_2 + k_1\dot{\boldsymbol{\sigma}}(\mathbf{e}_1) + g\mathbf{u}_3 - \ddot{\mathbf{p}}_d}{\left\|k_1^2k_2\mathbf{e}_2 + k_1\dot{\boldsymbol{\sigma}}(\mathbf{e}_1) + g\mathbf{u}_3 - \ddot{\mathbf{p}}_d\right\|}$
$T = \mathbf{r}_{3d}^T\mathbf{r}_3T_d$

Calculate $\ddot{\mathbf{e}}_1$ and $\dot{\mathbf{e}}_2$
$\ddot{\mathbf{e}}_1 = -\frac{T}{m}\mathbf{R}\mathbf{u}_3 + g\mathbf{u}_3 - \ddot{\mathbf{p}}_d$
$\dot{\mathbf{e}}_2 = \dot{\boldsymbol{\sigma}}(\mathbf{e}_1) + \frac{1}{k_1}\ddot{\mathbf{e}}_1$

Calculate $\mathbf{R}_d$, $\dot{\mathbf{R}}_d$ and $\boldsymbol{\omega}_d$
$\mathbf{R}_d = \left[-\frac{S(\mathbf{r}_{3d})^2\mathbf{p}'_d}{\|S(\mathbf{r}_{3d})^2\mathbf{p}'_d\|}\ \frac{S(\mathbf{r}_{3d})\mathbf{p}'_d}{\|S(\mathbf{r}_{3d})\mathbf{p}'_d\|}\ \mathbf{r}_{3d}\right]$
$\dot{\mathbf{R}}_d = \frac{\partial R_d}{\partial\mathbf{r}_{3d}}\dot{\mathbf{r}}_{3d} + \frac{\partial R_d}{\partial\mathbf{p}'_d}\mathbf{p}''_d\dot{\gamma}$
$\mathbf{S}(\boldsymbol{\omega}_d) = \mathbf{R}_d^T\dot{\mathbf{R}}_d \rightarrow \boldsymbol{\omega}_d = [S(\boldsymbol{\omega}_d)_{23}, S(\boldsymbol{\omega}_d)_{31}, S(\boldsymbol{\omega}_d)_{12}]$

Calculate $\dot{\omega}_{3c}$, $\mathbf{e}_4$, $\dot{\boldsymbol{\omega}}_c$ and $\boldsymbol{\tau}$
$\dot{\omega}_{3c} = -l_2\left(\omega_3 - \omega_{3d} + l_1\mathbf{r}_{2d}^T\mathbf{r}_1\right) + \dot{\omega}_{3d} - l_1\frac{d}{dt}\left(\mathbf{r}_{2d}^T\mathbf{r}_1\right)$
$\mathbf{e}_4$ *from Eq. 15*
$\dot{\boldsymbol{\omega}}_c$ *from Eq. 16*
$\boldsymbol{\tau} = \mathbf{J}\dot{\boldsymbol{\omega}}_c + \mathbf{S}(\boldsymbol{\omega})\mathbf{J}\boldsymbol{\omega}$

Obtain real torque inputs
$\tau_\phi = \tau_1,\ \tau_\theta = -\tau_2,\ \tau_\psi = -\tau_3$

return $T, \tau_\phi, \tau_\theta, \tau_\psi$

it is known that this system is fully linearisable with this set of outputs. Thus, the Full State Feedback Linearisation problem can be solved now for the system defined in Eqs. 27, 36 and 37.

To solve the Full State *FL* problem, the vector function $\boldsymbol{\Psi}(\mathbf{x}_e)$ that transforms the states of the system to a set of linear states is obtained as

$$\begin{bmatrix}\boldsymbol{\xi}_{1i}\\ \boldsymbol{\xi}_{2i}\\ \boldsymbol{\eta}_{1i}\\ \boldsymbol{\eta}_{2k}\end{bmatrix} = \boldsymbol{\Psi}(\mathbf{x}_e) = \begin{bmatrix}L_f^{i-1}h_1(\mathbf{x}_e)\\ L_f^{i-1}h_2(\mathbf{x}_e)\\ L_f^{i-1}h_3(\mathbf{x}_e)\\ L_f^{k-1}h_4(\mathbf{x}_e)\end{bmatrix}, \tag{38}$$

for $i \in \{1, 2, 3, 4\}$ and $k \in \{1, 2\}$. Where x_e is the set of 14 states of the extended system, $\boldsymbol{\xi}_{1i}$ and $\boldsymbol{\xi}_{2i}$ are the linear states that are required to converge to zero (h_1 and h_2 subsystems), $\boldsymbol{\eta}_{1i}$ and $\boldsymbol{\eta}_{2k}$ are the linear states that have to be regulated to control the position or velocity on the path (h_3 subsystem) and the *yaw* angle (h_4 subsystem). Moreover, $L_f^m h_n(\mathbf{x_e})$ refers to the mth Lie derivative of $h_n(\mathbf{x_e})$ with respect to f.

It is important to mention that $\boldsymbol{\Psi}(\mathbf{x}_e)$ is a diffeomorphism. That is, a derivable function defined in a certain region whose inverse exists and is derivable as well.

Once the transformation of Eq. 38 is applied to the extended system, the resulting system is

$$\begin{aligned}
\dot{\xi}_{11} &= \xi_{12}\\
\dot{\xi}_{12} &= \xi_{13}\\
\dot{\xi}_{13} &= \xi_{14}\\
\dot{\xi}_{14} &= \alpha_1(\boldsymbol{\xi}, \boldsymbol{\eta}) + \beta_1(\boldsymbol{\xi}, \boldsymbol{\eta})\,\mathbf{u}\\
\dot{\xi}_{21} &= \xi_{22}\\
\dot{\xi}_{22} &= \xi_{23}\\
\dot{\xi}_{23} &= \xi_{24}\\
\dot{\xi}_{24} &= \alpha_2(\boldsymbol{\xi}, \boldsymbol{\eta}) + \beta_2(\boldsymbol{\xi}, \boldsymbol{\eta})\,\mathbf{u}\\
\dot{\eta}_{11} &= \eta_{12}\\
\dot{\eta}_{12} &= \eta_{13}\\
\dot{\eta}_{13} &= \eta_{14}\\
\dot{\eta}_{14} &= \alpha_3(\boldsymbol{\xi}, \boldsymbol{\eta}) + \beta_3(\boldsymbol{\xi}, \boldsymbol{\eta})\,\mathbf{u}\\
\dot{\eta}_{21} &= \eta_{22}\\
\dot{\eta}_{22} &= \alpha_4(\boldsymbol{\xi}, \boldsymbol{\eta}) + \beta_4(\boldsymbol{\xi}, \boldsymbol{\eta})\,\mathbf{u}
\end{aligned} \tag{39}$$

where the vector formed by each $\boldsymbol{\alpha}$ function is calculated as shown in Eq. 40 and the $\boldsymbol{\beta}$ functions conform the (4x4) decoupling matrix of the system, which is obtained in

Eq. 41.

$$\boldsymbol{\alpha}(\boldsymbol{\xi},\boldsymbol{\eta}) = \begin{bmatrix} \alpha_1(\boldsymbol{\xi},\boldsymbol{\eta}) \\ \alpha_2(\boldsymbol{\xi},\boldsymbol{\eta}) \\ \alpha_3(\boldsymbol{\xi},\boldsymbol{\eta}) \\ \alpha_4(\boldsymbol{\xi},\boldsymbol{\eta}) \end{bmatrix} = \begin{bmatrix} L_f^4 h_1 \\ L_f^4 h_2 \\ L_f^4 h_3 \\ L_f^2 h_4 \end{bmatrix} \tag{40}$$

$$\boldsymbol{\beta}(\boldsymbol{\xi},\eta) = \begin{bmatrix} \beta_1(\boldsymbol{\xi},\boldsymbol{\eta}) \\ \beta_2(\boldsymbol{\xi},\boldsymbol{\eta}) \\ \beta_3(\boldsymbol{\xi},\boldsymbol{\eta}) \\ \beta_4(\boldsymbol{\xi},\boldsymbol{\eta}) \end{bmatrix} = \begin{bmatrix} L_g L_f^3 h_1 \\ L_g L_f^3 h_2 \\ L_g L_f^3 h_3 \\ L_g L_f h_4 \end{bmatrix} \tag{41}$$

It can be observed that the resulting system of Eq. 39 is composed by four canonical controllable subsystems. Thus, if the inputs of the system (u_{1-4}) are chosen to be

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \beta_1(\boldsymbol{\xi},\boldsymbol{\eta})^{-1}\left[v_1 - \alpha_1(\boldsymbol{\xi},\boldsymbol{\eta})\right] \\ \beta_2(\boldsymbol{\xi},\boldsymbol{\eta})^{-1}\left[v_2 - \alpha_2(\boldsymbol{\xi},\boldsymbol{\eta})\right] \\ \beta_3(\boldsymbol{\xi},\boldsymbol{\eta})^{-1}\left[v_3 - \alpha_3(\boldsymbol{\xi},\boldsymbol{\eta})\right] \\ \beta_4(\boldsymbol{\xi},\boldsymbol{\eta})^{-1}\left[v_4 - \alpha_4(\boldsymbol{\xi},\boldsymbol{\eta})\right] \end{bmatrix}, \tag{42}$$

the system results in a pure multiple integrator of r_i-order for each virtual output h_i, where r_i is the relative degree. Then, the virtual input, v_i, is obtained by applying a state feedback control with the respective linearised states as shown in Eq. 43. The feedback gains K_{ij} can be calculated using the pole-placement method.

$$\begin{aligned} v_1 &= \xi_{11}\,K_{11} + \xi_{12}\,K_{12} + \xi_{13}\,K_{13} + \xi_{14}\,K_{14} \\ v_2 &= \xi_{21}\,K_{21} + \xi_{22}\,K_{22} + \xi_{23}\,K_{23} + \xi_{24}\,K_{24} \\ v_3 &= (\eta_{11} - \eta_{11}^{ref})\,K_{31} + (\eta_{12} - \eta_{12}^{ref})\,K_{32} \\ &\quad + \eta_{13}\,K_{33} + \eta_{14}\,K_{34} \\ v_4 &= (\eta_{21} - \eta_{21}^{ref})\,K_{41} + \eta_{22}\,K_{42} \end{aligned} \tag{43}$$

The control structure of the *FL* controller is shown in Fig. 5. It is an IGC structure as the *Backstepping* algorithm. As shown, there is a block that transforms the states of the extended system ($\mathbf{x}, \zeta_1, \zeta_2$) into the set of linear states ($\boldsymbol{\xi}, \boldsymbol{\eta}$). Then, another block calculates the virtual inputs ($\mathbf{v}$) from these linear states. Later, the virtual inputs are transformed into the inputs of the extended system ($\mathbf{u}$). And finally, the first input (u_1) is integrated twice to obtain the total thrust applied on the quadrotor (T), which is the real input of the system. Note that this scheme does not include the path as an external reference, since it must be embedded in the algorithm calculations.

The implementation of this *Feedback Linearisation* control is stated in Algorithm 2. This algorithm requires the full state of the extended system ($\mathbf{x}_e$), the radius of the circumference (A), the desired velocity of the vehicle (V_{path}) and the feedback gains of the linear controller (K_{ij}). It returns the four inputs of the quadrotor. The algorithm starts by an axis transformation of the velocities to the world frame. Later, the outputs of the system are obtained by means of Eq. 36. Note that the arctan is implemented using the *atan2* function. Next, the references of the linear states (η_{11}^{ref}, η_{12}^{ref} and η_{21}^{ref}) are calculated. These define the experiment references and are detailed in the Section 5. Note that it is ensured that η_{21}^{ref}, the *yaw* reference, satisfies the condition $|\ \eta_{21}^{ref} - \psi\ | \leqslant \pi$, since it is expected that the vehicle moves the smallest angle towards its reference *yaw* angle. Final steps are the calculation of the linear states ($\boldsymbol{\xi}$, $\boldsymbol{\eta}$), $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (38, 40 and 41), to obtain the virtual inputs (43), and to calculate the inputs used to control the system (42). It is important to note that the input of the total thrust of the quadrotor is obtained by double integrating the u_1 input of the extended system.

Algorithm 2 Feedback Linearization.

Require: $\mathbf{x}_e := \{x, y, z, u, v, w, p, q, r, \phi, \theta, \psi, \zeta_1, \zeta_2\}$, A, V_{path}, K_{ij}

Transform velocities axis:

$\left[u^W\ v^W\ w^W\right]^T = \mathbf{R}\,[u\ v\ w]^T$

Calculate outputs:

$h_1 = x^2 + y^2 - A^2$

$h_2 = Z - f_z(\mathbf{x}_e)$

$h_3 = A\,\text{atan2}(y, x)$

$h_4 = \psi$

Calculate linear state references:

$\eta_{11}^{ref} = 0\ \ (K_{31} = 0)$

$\eta_{12}^{ref} = V_{path}$

$\eta_{21}^{ref} = h_3/A + \pi/2$

$\rightarrow$ ensure that $\left|\eta_{21}^{ref} - \psi\right| \leqslant \pi$

Calculate linear states ($\boldsymbol{\xi}$, $\boldsymbol{\eta}$), $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$

$\{\boldsymbol{\xi}_{1i}, \boldsymbol{\xi}_{2i}, \boldsymbol{\eta}_{1i}, \boldsymbol{\eta}_{2k}\} := \boldsymbol{\Psi}(\mathbf{x}_e)$

$\alpha_i = L_f^m h_i$

$\beta_i = L_g L_f^{m-1} h_i$

$\rightarrow i \in \{1,\ 2,\ 3,\ 4\}, k \in \{1,\ 2\}$

Calculate virtual inputs

$\{v_1,\ v_2,\ v_3,\ v_4\}$ *from Eq. 43*

Calculate inputs of the extended system

$u_i = \beta_i^{-1}\,[v_i - \alpha_i]$

$\rightarrow i \in \{1,\ 2,\ 3,\ 4\}$

Obtain real inputs

$\dot{\zeta}_2 = u_1,\ \dot{\zeta}_1 = \zeta_2,\ T = \zeta_1,$

$\tau_\phi = u_2,\ \tau_\theta = u_3,\ \tau_\psi = u_4$

return $T, \tau_\phi, \tau_\theta, \tau_\psi$

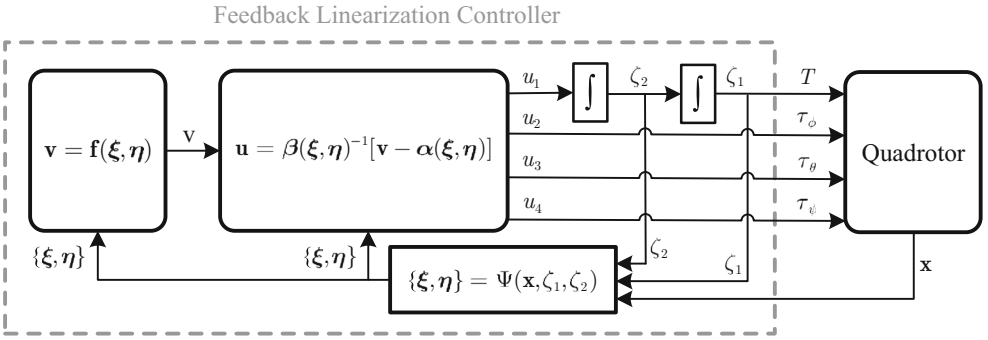

Fig. 5 Feedback Linearization control structure

4.4 Three-Dimensional NLGL

NLGL, as mentioned in Section 3.4, is a geometric PF algorithm based on the Virtual Target Point (VTP) concept. VTP is a target reference point placed on the desired path, which is updated periodically. The *NLGL* algorithm obtains the VTP as the point on the path at a predefined distance L from the vehicle, where L is a scalar constant parameter that affects the performance of the *NLGL* controller. Once the VTP is obtained, the algorithm calculates the needed *yaw* angle to face the vehicle towards the VTP on the path. Since the vehicle is required to move at a constant predefined speed, it will end up moving to the path.

From the algorithm description it can be seen that it is designed to work only in the two-dimensional space given by the xy plane. For this reason, as the two previous algorithms operate in three dimensions and the objective of this paper is to compare the algorithms in the same conditions, *NLGL* algorithm presented in [78] has been adapted to operate in 3D as well. To this end, the *3D-NLGL* algorithm has been stated in Algorithm 3.

The Algorithm 3 requires the current position $\mathbf{p}$ of the vehicle and its *yaw* angle (ψ), as well as the desired velocity of the vehicle on the x body axis (V_{path}) and the scalar parameter of the guidance law (L). It returns the references for the velocities in the x and y axis, the altitude and the *yaw* angle of the quadrotor. In this algorithm, γ_{prev} represents the value of $\gamma_{d_{\min}}$ saved from the previous execution of the algorithm, where $\gamma_{d_{\min}}$ is the point on the path, given by the value of γ, that is at a minimum distance to the location of the vehicle. The value of γ_{prev} is initialized to 0 in the first execution of this algorithm.

As seen in the algorithm, the first step after the initialization of the variables is to calculate the point on the path that is at a minimum distance from the vehicle, given by $\gamma_{d_{\min}}$, and the value of this distance. Then, if the distance to $\gamma_{d_{\min}}$ is larger than L, the VTP is defined as $\gamma_{d_{\min}}$, making the vehicle move directly towards the path. When the vehicle is closer to the path, the algorithm will find the first point on the path, starting from $\gamma_{d_{\min}}$, that is at an L distance from the vehicle, and will set it as the VTP. The fact that it starts to search the VTP from the point at a minimum distance ($\gamma_{d_{\min}}$) going forwards, ensures that the vehicle moves always forwards on the path.

The next step of this algorithm is to calculate the references, which are the outputs of the controller. This is important to ensure the correct adaptation of the algorithm from 2D to 3D. First of all, similarly to the *2D-NLGL* algorithm, the *yaw* reference angle is calculated as the angle

Algorithm 3 Three-dimensional NLGL.

Require: $\mathbf{p} : (x,\ y,\ z),\ \psi,\ L,\ V_{path},$
$\mathbf{p}_d(\gamma) := [x_d(\gamma),\ y_d(\gamma),\ z_d(\gamma)]^T,\ \gamma_f$

Initialize (First execution):
$\gamma_{prev} = 0$

Calculate $d_{\min}, \gamma_{d_{\min}}$:
$d_{\min} := \min_{\gamma} (\|\mathbf{p} - \mathbf{p}_d(\gamma)\|) \mid \gamma \in [\gamma_{prev};\ \gamma_f]$
$\gamma_{d_{\min}} := \operatorname{argmin}_{\gamma} (\|\mathbf{p} - \mathbf{p}_d(\gamma)\|) \mid \gamma \in [\gamma_{prev};\ \gamma_f]$

Calculate γ_{VTP}:
if $d_{\min} > L$ **then**
    $\gamma_{VTP} := \gamma_{d_{\min}}$
else
    $\gamma_{VTP} := \operatorname{argmin}_{\gamma} (|\|\mathbf{p} - \mathbf{p}_d(\gamma)\| - L|) \mid \gamma \in [\gamma_{d_{\min}};\ \gamma_f]$
end if

Calculate references:
$\psi_{ref} := \text{atan2}(y_d(\gamma_{VTP}) - y,\ x_d(\gamma_{VTP}) - x)$
$\rightarrow$ ensure that $\left|\psi_{ref} - \psi\right| \leqslant \pi$
$z_{ref} := z_d(\gamma_{d_{\min}})$
$v_{ref} := 0$
$u_{ref} := \frac{V_{path} d_{VTP,xy}}{L}$

return $\psi_{ref}, z_{ref}, v_{ref}, u_{ref}$

that would face the vehicle towards the VTP, projected in the xy plane of the vehicle. Then, as in the *Feedback Linearisation* controller, it is ensured that ψ_{ref} satisfies the condition $\mid \psi_{ref} - \psi \mid \leqslant \pi$. The altitude reference Z_{ref} is obtained as the desired Z in the point of minimum distance $\gamma_{d_{\min}}$.

As in the two-dimensional algorithm, the velocity reference v_{ref} in the y body coordinate is set to zero, as no movement is required on this coordinate. In the *2D-NLGL* algorithm the velocity reference u_{ref} in the x body coordinate has a constant value (V_{path}). Considering the vertical distance to the VTP, in this 3D algorithm the value of u_{ref} is set proportional to $d_{VTP,xy}$, that is the projection on the xy plane of the distance L to the VTP. Line 11 of the algorithm calculates this reference. It can be seen that when the VTP is on the xy plane $u_{ref} = V_{path}$.

The outputs of this controller are the u_{ref} and v_{ref} velocity references, the *yaw* reference (ψ_{ref}) and the altitude reference (z_{ref}). An autopilot is needed, implemented as a set of *Proportional-Integral-Derivative* (*PID*) controllers designed to control the attitude, altitude and velocities. This autopilot has been designed following a cascade control structure formed by a velocity controller (u and v) and an attitude & altitude controller (ϕ, θ, ψ and z), as illustrated in Fig. 6. In this figure, the complete control scheme including the PF controller, the autopilot and the plant is presented. *PID* controllers are designed by means of the pole placement technique using the linearised plant. The *PID* parameters of the autopilot are given in Table 3.

It is important to mention that, with an SGC structure, the path following controller's behaviour will always be affected by the performance of the inner-loop controller. Other techniques, such as *LQR*, could be applied to design the autopilot controller with different results. However, the *PID* controllers, which do not require the full state estimation, have been chosen by their simplicity which is aligned with the simplicity of the geometric algorithms. This allows to make a comparison of the control algorithms (*Backstepping* and *Feedback linearisation*) with the simplest PF algorithms.

Table 3 Constants of the *PID* controllers of the autopilot

	Kp	Ki	Kd
u	0.32	-	0.1
v	0.32	-	0.1
ϕ	2	1.1	1.2
θ	2	1.1	1.2
ψ	4	0.5	3.5
z	2	2.1	3.3

4.5 Three-Dimensional Carrot-Chasing

Carrot-Chasing is also a geometric algorithm based on the VTP concept. In the *CC* approach, the VTP is called carrot and is obtained as the point at constant length δ along the path from the point at a minimum distance to the vehicle. That is, δ is computed along the path arc length, and therefore it is not equivalent to the Euclidean distance computed in the *NLGL* algorithm. The vehicle, which is required to move at a constant speed, is faced to the VTP by changing its *yaw* angle, and thus it is said to chase the carrot.

Again, it can be noted that the algorithm is designed to operate in the two-dimensional space. Thus, in this paper it has been modified and adapted to work in the three-dimensional space. The resulting developed 3D approach is found in Algorithm 4. This algorithm is very similar to the *3D-NLGL* algorithm, since the structure and the operations to calculate the references are identical. However, in this case the VTP is obtained as the point on the path that is at a δ length from the $\gamma_{d_{\min}}$ point, where this length is computed along the path arc. Another difference between *CC* and

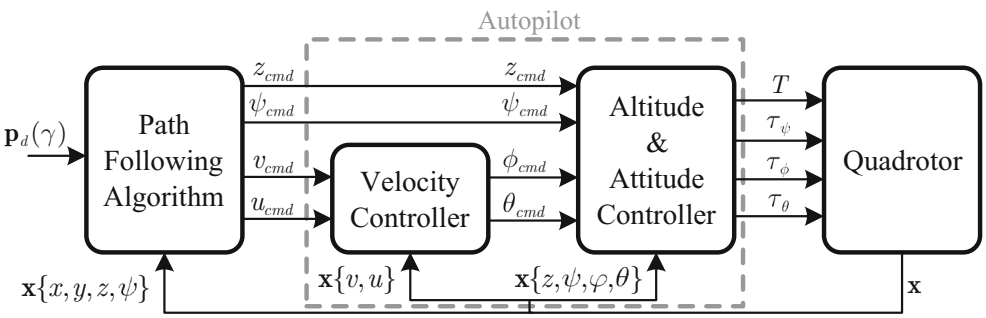

Fig. 6 SGC structure for geometric algorithms

Algorithm 4 Three-dimensional Carrot-Chasing.

Require: $\mathbf{p} : (x,\ y,\ z),\ \psi,\ \delta,\ V_{path},$
$\mathbf{p}_d(\gamma) := [x_d(\gamma),\ y_d(\gamma),\ z_d(\gamma)]^T,\ \gamma_f$

Initialize (First execution):
1: $\gamma_{prev} := 0$

Calculate $d_{\min}, \gamma_{d_{\min}}$:
2: $d_{\min} := \min_{\gamma} (\|\mathbf{p} - \mathbf{p}_d(\gamma)\|) \mid \gamma \in [\gamma_{prev};\ \gamma_f]$
3: $\gamma_{d_{\min}} := \operatorname{argmin}_{\gamma} (\|\mathbf{p} - \mathbf{p}_d(\gamma)\|) \mid \gamma \in [\gamma_{prev};\ \gamma_f]$

Calculate γ_{VTP}:
4: $\gamma_{VTP} := \operatorname{argmin}_{\gamma} \left(\left| \left(\int_{\gamma_{d_{\min}}}^{\gamma} \|\mathbf{p}_d'(\gamma)\|\, d\gamma \right) - L \right| \right) \mid \gamma \in [\gamma_{d_{\min}};\ \gamma_f]$

Calculate references:
5: $\psi_{ref} := \text{atan2}(y_d(\gamma_{VTP}) - y,\ x_d(\gamma_{VTP}) - x)$
$\rightarrow$ ensure that $\left|\psi_{ref} - \psi\right| \leqslant \pi$
6: $z_{ref} := z_d(\gamma_{d_{\min}})$
7: $v_{ref} := 0$
8: $u_{ref} := \frac{V_{path} d_{VTP,xy}}{d_{VTP}}$

return $\psi_{ref}, z_{ref}, v_{ref}, u_{ref}$

NLGL algorithms is that in *CC* the VTP is always calculated the same way, regardless of the distance from the UAV to the path.

The control structure used to apply the *3D-CC* is the same as the *3D-NLGL*, represented in the Fig. 6. Thus, the same autopilot has been used.

5 Algorithm Comparison Based on Simulation Results

In this section a set of simulation results comparing the four algorithms described previously is presented. The simulations have been performed on the *Quad-Sim* simulator stated in Section 4.1. The desired path is a helix defined by

$$\mathbf{p}_d(\gamma) = \begin{bmatrix} A\cos(\gamma) \\ A\sin(\gamma) \\ \gamma + 3 \end{bmatrix} \tag{44}$$

where A is the radius of the helix, which is 3 meters. Note that with this definition the path starts at 3 meters of altitude. Additionally, the vehicle is required to travel at a constant velocity V_{path} on the path and with a *yaw* angle tangent to the path.

To meet the velocity specification with the *Backstepping* controller, the desired evolution of γ has been set as defined in Eq. 45. The *yaw* requirement is assured by defining the rotational matrix as in Eq. 18 and including the PD-like controller for the angular acceleration on the z axis as defined in Eq. 19.

$$\dot{\gamma}_d = \frac{V_{path}}{A} \tag{45}$$

Assuming that the vehicle's orientation is tangent to the path, γ can be obtained subtracting $\pi/2$ to the *yaw* reference (ψ_{ref}). Thus, function $f_z(\mathbf{x})$ (Eq. 36 of *Feedback Linearisation* algorithm) becomes

$$f_z(\mathbf{x}) = \gamma + 3 = \psi_{ref} - \frac{\pi}{2} + 3 \tag{46}$$

In the *FL* controller the desired velocity requirement is met by making the reference of the first derivative of h_3 (η_{12}^{ref}) equal to V_{path}. The feedback gain K_{31}, which regulates the position of the vehicle along the path, is set to zero. Hence, h_3 is only in charge of controlling the velocity of the vehicle along the path. On the other hand, to follow the *yaw* specification, the reference for h_4 is calculated as the path tangent angle on the closest point of the path to the vehicle, as seen on line 8 of the Algorithm 2.

For the *3D-NLGL* and *3D-CC* algorithms, the velocity and *yaw* specifications are accomplished by construction.

The control parameters for each of the four algorithms have been tuned as those that achieve the least mean absolute error (MAE) in terms of distance to the path error. This MAE has been evaluated in the simulation platform presented on the next Section (5.1) These parameters are presented in Table 4. For the geometric algorithms, as they only have one parameter to tune, a parameter sweep was performed in order to find the value which procures the minimum distance MAE for the defined path. It is important to remark that those parameters depend on the velocity of the vehicle and the path shape. Regarding the FL algorithm, the constants that control each of the four linear subsystems (xy-position, altitude, path velocity and *yaw*) were tuned separately minimizing the error of each variable. It was done by means of an iterative redesign through the pole

Table 4 Control parameters for each algorithm

Algorithm	Parameters
Backstepping	$k_1 = 2, k_2 = 1, k_3 = 50, k_4 = 2, k_\gamma = 50,$ $l_1 = 10, l_2 = 2, p_{\max} = 1, k_f = 10$
Feedback Linearization	$k_{11} = -18.75, k_{12} = -40, k_{13} = -32.25, k_{14} = -10,$ $k_{21} = -1\,250, k_{22} = -1\,650, k_{23} = -435, k_{24} = -36,$ $k_{31} = 0, k_{32} = -104, k_{33} = -124, k_{34} = -21,$ $k_{41} = -29, k_{42} = -10$
3D-NLGL	$L = 0.59$
3D Carrot-Chasing	$\delta = 0.59$

placement technique, assuring stability on each iteration. Once the constants of each subsystem were obtained, it was verified that the complete system still behaved with minimum path following error and that it was stable. In the BS algorithm, the stability of the controlled system is very sensitive to changes in the control parameters (k_{1-4}). As they were difficult to tune, they were empirically set to make the system stable. The rest of the parameters of this algorithm were obtained by means of a parameter sweep search.

Three types of results are reported in this section. First, on the steady state regime, when the vehicle has converged to the path. Next, on the transient regime, changing the initial state conditions. Last, with time-varying wind disturbances.

5.1 Simulation Platform

The simulation platform used in this paper is the *Quad-Sim* simulator [38]. An open source MATLAB tool developed by a group of students from the Drexel University that allows the modelling and simulation of quadcopter vehicles for control system design.

This platform is built by Simulink models of the dynamic system behaviour. It implements the non-linear dynamic equations of the quadrotor vehicle, described in Section 4.1, as well as the dynamics of the motors and other aerodynamic effects, such as drag forces. The motors are modelled as a first order system with a saturation on the angular velocity. The values of some of the most important parameters of the employed simulation model are detailed in Table 5.

The authors of the present paper considered this tool suitable to apply and compare the developed algorithms since it implements a conventional non-linear and complete dynamic model of the quadrotor vehicle, which has been commonly described in the literature [13, 52]. Moreover, simulation provides the ground truth and repeatability necessary to compare PF algorithms.

Table 5 Values of the simulation model parameters

Parameter	Value	Unit
m	1023	g
J_{xx}	0.0095	$kg\,m^2$
J_{yy}	0.0095	$kg\,m^2$
J_{zz}	0.0186	$kg\,m^2$
c_T	$1.4865e^{-7}$	$N\,rpm^2$
c_Q	$2.9250e^{-9}$	$N\,m\,rpm^2$
d	0.2223	m
J_m	$3.7882e^{-6}$	$kg\,m^2$

5.2 Steady State Regime

This section shows results about the steady state regime of the vehicle's response with each controller. Steady state refers to the controller's error being constant. The performance of the four algorithms for one full lap of the helix is compared in Table 6. The columns relate to: the time to travel one lap, the mean of the distance to the path ($\overline{d}$), the mean absolute *yaw* error, the mean velocity of the quadrotor and the computational effort of the algorithm. The path distance is the minimum distance between the path and the vehicle, the mean absolute *yaw* error is calculated from the error between the vehicle's *yaw* angle and the path tangent angle and the computational effort is a dimensionless parameter that represents the normalized computation time of each algorithm.

The computational effort was calculated as follows: A discrete time simulation with a time step of 1ms was performed for each of the four algorithms and the execution time of each block of the simulink model were monitored. The runtime dedicated to execute the calculations of the control part were divided by the runtime needed for the calculations of the dynamics of the system for each algorithm. Assuming that the time dedicated to compute the dynamics of the quadrotor is similar on each execution, the obtained quotient is considered a representative value to evaluate the computational effort. Finally, result was normalized, setting to 1 the lowest one which corresponds to the *NLGL* algorithm. The implementation of the algorithms was not optimized in terms of computation time. Nevertheless, none of them includes especially intensive calculations (such as solving optimization problems or matrix operations). Thus, they are subject to slight changes on the computation effort term.

The three-dimensional trajectory for one lap of the helix in steady state regime using the *3D-NLGL* algorithm is shown in Fig. 7. The performance of this algorithm in path distance is poor compared to the other algorithms where the vehicle stabilized closer to the path.

5.3 Transient Regime

In these simulations the initial state conditions are modified to observe the transient behaviour of the algorithms. In particular, the effects of changing the initial x-coordinate of the vehicle and the effects of varying its initial *yaw* are analyzed.

In the simulations where the initial x-coordinate position changes, the quadrotor starts on the position given by $x = \{0.5k \mid k = 1..12\}$, $y = 0$ and $z = 3$. The initial *yaw* angle is 90 degrees in the case of the *BS* and *FL* controllers whereas in the geometric algorithms the vehicle faces the VTP. This orientations result in a minimum initial *yaw* error. The initial

Table 6 Results for one lap in steady state regime

	time (s)	$\overline{d}$ (m)	$\overline{\lvert e_\psi \rvert}$ (deg)	$\overline{\lVert \mathbf{v} \rVert}$ (m/s)	Computational effort
Backstepping	44.6306	0.0016	1.6094	0.4451	57.3895
Feedback Linearization	40.7400	0.0078	3.0522	0.4874	1.4877
3D-NLGL	40.3800	0.0198	2.6953	0.4897	1
3D-Carrot-Chasing	40.3800	0.0194	2.6949	0.4897	1.0147

vehicle's linear velocities, angular velocities, accelerations, roll and pitch angles are zero.

Figure 8 shows four performance indicator graphs, for each algorithm and for each initial position. First, the time the quadrotor takes to converge to the path. Convergence condition is defined as the vehicle remaining at path distance less than 10 times the stable state regime error. Second plot shows the accumulated distance to the path during this convergence period. Note that the periods are different for each algorithm, since their stabilization time are different. The third graph, shows a parameter for the control effort, i.e. the mean of the absolute value of the control action derivative. Note that the control action of motors (u_{mi}) is given in percentage. Finally, the convergence position on the path given by γ is represented, where the magnitude of γ is given in number of radius along the path (A). The mathematical definition of these indicators is stated in Eqs. 47–50, where t_{conv} is the stabilization time, d is the distance to the path, d_{ss} is the distance error on the steady state regime, d_{int} is the integral of the path distance on the convergence time, c_{eff} is the control effort term, u_{mi} is the control action of the ith motor and γ_{conv} is the convergence position given by the virtual arc parameter.

$$t_{conv} : d < 10 d_{ss} \; \forall \, t > t_{conv} \tag{47}$$

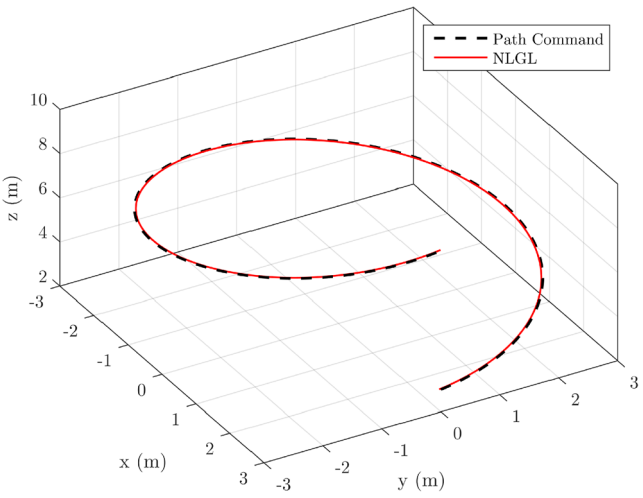

Fig. 7 3D trajectory for one lap of the helix in steady state regime: *3D-NLGL* algorithm

$$d_{int} = \int_{t_{init}}^{t_{conv}} d(t) dt \tag{48}$$

$$c_{eff} = \frac{\left(\overline{\left| \frac{du_{m1}}{dt} \right|} + \overline{\left| \frac{du_{m2}}{dt} \right|} + \overline{\left| \frac{du_{m3}}{dt} \right|} + \overline{\left| \frac{du_{m4}}{dt} \right|} \right)}{4} \tag{49}$$

$$\gamma_{conv} = \gamma(t_{conv}) \tag{50}$$

Figures 9 and 10 show the three-dimensional trajectory when the initial x-coordinate is 0.5m and 6m, respectively, comparing the response of the four algorithms for one lap. In both figures, circles indicate the convergence point of each algorithm. A solid line represents the convergence trail, and a dotted line where the vehicle has converged.

The simulation results changing the initial *yaw* angle are shown in Fig. 11. In these simulations the vehicle starts at the initial point of the path ($x = 3, y = 0, z = 3$) and the *yaw* orientation varies from 0 to 180 degrees in steps of 22.5 in each simulation. The rest of the initial state conditions are identical to the previous simulations. Furthermore, the plots represented in Fig. 11 are equivalent to the ones found in Fig. 8. Since the convergence criterion

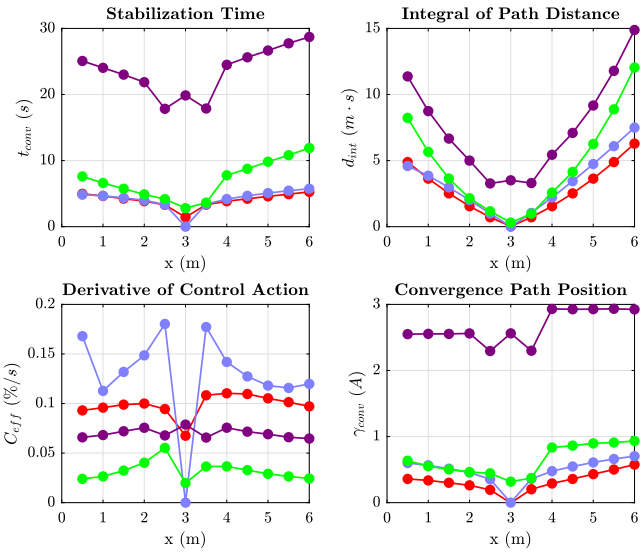

Fig. 8 Simulation results varying the initial x position from 0.5 to 6 meters: *Backstepping* (red), *Feedback Linearisation* (blue), *3D-NLGL* (purple) and *3D Carrot-Chasing* (green)

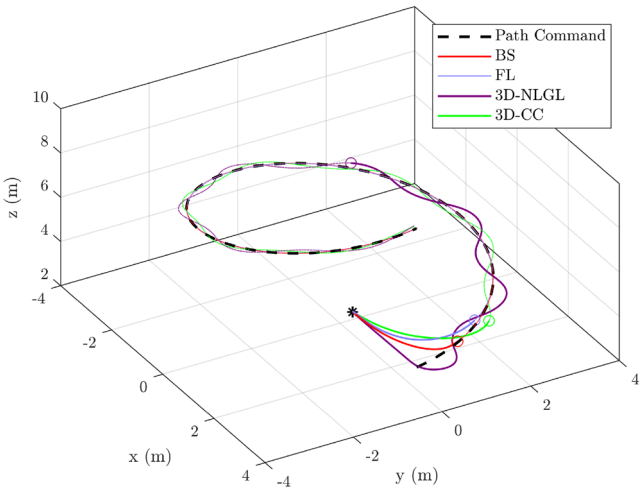

Fig. 9 3D trajectory comparing the four algorithms for one lap of the helix. Initial position: $x = 0.5$, $y = 0$, $z = 3$

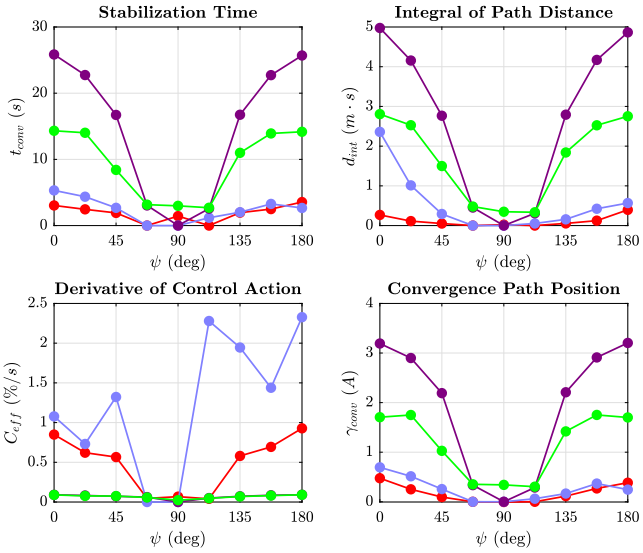

Fig. 11 Simulation results varying the initial *yaw* angle from 0 to 180 degrees: *Backstepping* (red), *Feedback Linearisation* (blue), *3D-NLGL* (purple) and *3D Carrot-Chasing* (green)

only takes into account the distance to the path, in some cases it is considered that the quadrotor has converged to the path while there is still significant *yaw* error.

Once more, the three-dimensional trajectory plots comparing the behaviour of each algorithm are presented in Figs. 12 and 13, for the extreme cases of the initial *yaw* angle. The convergence point, represented with a circle, divide the convergence trail (solid line) and the rest of the trajectory (dotted line).

5.4 Wind Disturbance

In the real world, UAVs deal with diverse environmental influences, such as wind disturbances. Thus, it is of utmost importance that path following algorithms show robustness to these external disturbances. The performance of the implemented algorithms in the presence of time-varying wind is analysed in this section.

To carry out these simulations a wind force has been created. Both its magnitude and direction changes randomly, ranging from 0 to 1 Newton and from $\pi/8$ to $3\pi/8$ radians (i.e. north-east direction), respectively. The performance of each of the four algorithms following the helix path with this wind disturbance force is shown in Fig. 14. In this simulations the vehicle starts from the initial point of the path at hover conditions (i.e. zero linear and angular velocities), and it is requested to follow the path for 100 seconds.

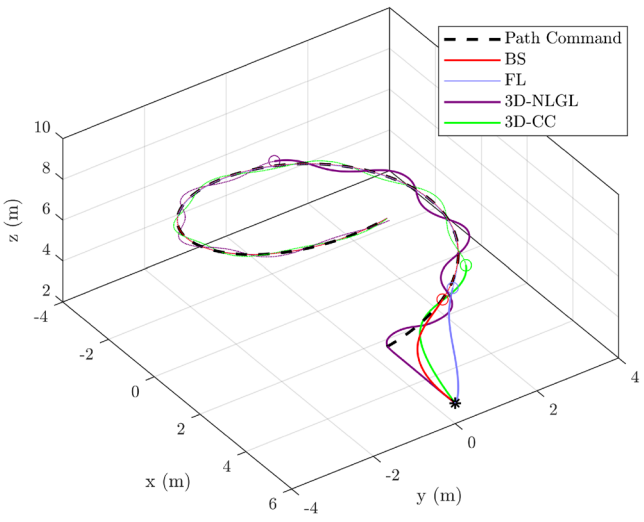

Fig. 10 3D trajectory evolution comparing the four algorithms for one lap of the helix. Initial position: $x = 6$, $y = 0$, $z = 3$

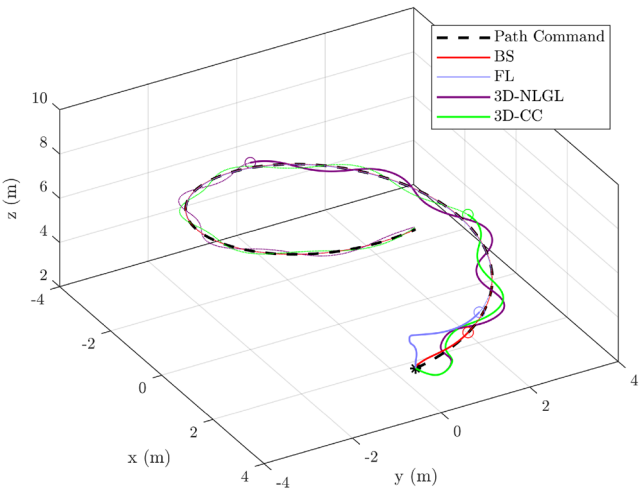

Fig. 12 3D trajectory evolution comparing the four algorithms for one lap of the helix. Initial position: $x = 3$, $y = 0$, $z = 3$. Initial yaw = 0

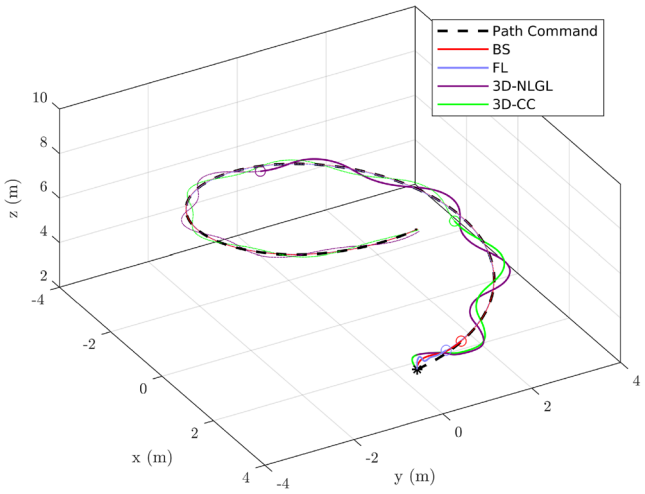

Fig. 13 3D trajectory evolution comparing the four algorithms for one lap of the helix. Initial position: $x = 3$, $y = 0$, $z = 3$. Initial yaw = 180

The first and second plot of Fig. 14 show the wind magnitude and direction, respectively, changing every 10 seconds. The third plot shows the evolution of the path distance error for each algorithm during the 100 seconds experiment.

The results of the simulation experiment of Fig. 14 are summarized in Table 7: the average path distance error, the average *yaw* error, the average velocity and the path travelled (given by the virtual arc γ), for each algorithm.

5.5 Discussion

From the steady state regime simulation results, in Table 6, it can be observed that the *Backstepping* algorithm is the

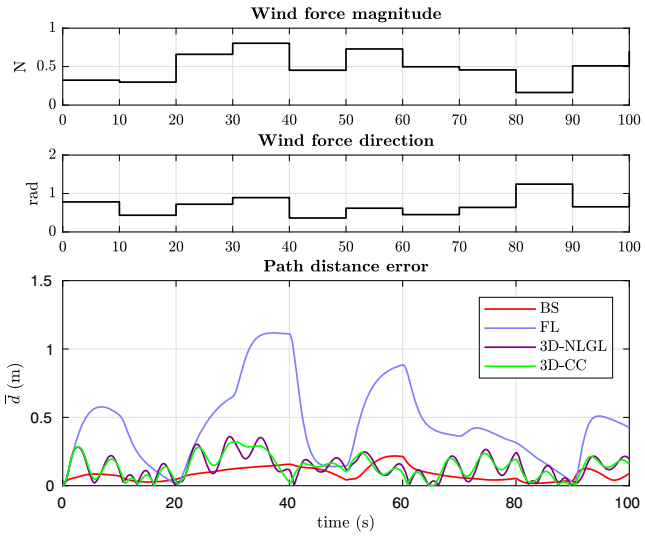

Fig. 14 Path distance error against wind disturbances

Table 7 Results for 100 seconds in time-varying wind conditions

	$\overline{d}\ (m)$	$\overline{\lvert e_\psi \rvert}\ (deg)$	$\overline{\lVert \mathbf{v} \rVert}\ (m/s)$	*Path traveled* (γ)
Backstepping	0.0859	2.5096	0.3935	12.3731
Feedback Linearisation	0.4491	0.8709	0.1761	2.7191
3D-NLGL	0.1396	14.9657	0.4381	13.6861
3D Carrot-Chasing	0.1377	14.0547	0.4385	13.8209

one that achieves the best performance in terms of distance to the path, which is usually the most important parameter. *BS* presents a little more than 1.5 mm of error, compared to the 8 mm of the *Feedback Linearisation* algorithm or the almost 2 cm of the geometric algorithms. However, the *BS* algorithm presents a very large computational effort in comparison to the other algorithms, which obtain similar values of this indicator. Another drawback of the *BS* algorithm is that it reduces the cruise velocity in order to achieve this great precision. This is observed in the higher time that it takes to accomplish one lap of the helix. Regarding the *yaw* error, again the *Backstesping* algorithm has the best response with a mean *yaw* error of 1.6. The geometric algorithms have a larger yaw error than *BS* because these algorithms are not designed to keep the vehicle tangent to the path but to face it to the VTP. *FL* algorithm shows a *yaw* error even larger than the geometric algorithms, although it is designed to be tangent to the path.

Regarding the transient regime results, in the simulations where the initial x-coordinate changes, the *3D-NLGL* algorithm presents the worst transient behaviour. It has wide oscillations along the path that increase its convergence time. The *3D Carrot-Chasing* algorithm also presents an oscillating performance. However, the oscillations are smaller, as can be noticed from the 3D plots of Figs. 9 and 10. This performance distinction between the two geometric algorithms is due to the way they approach to the path. The *3D-NLGL* algorithm moves directly to the minimum distance point on the path ($\gamma_{d_{\min}}$), while the *3D-CC* moves always to a δ distance from this point. This slight difference becomes significant on the final performance. From our experience, these oscillations on the geometric algorithms, and thus the convergence time, can be reduced by increasing the geometric control parameters (L and δ). However, this results in an increment of the path distance error too.

The modified initial x-coordinate simulations also reflect that the control-oriented algorithms (*BS* and *FL*) obtain very similar stabilization times. Neverthelles, *FL* results in higher path distance errors because it converges to a further point on the path, as evidenced by the convergence path position plot (Fig. 8).

The control-oriented algorithms, especially the *FL* algorithm, make a higher control effort to remain on the path (Fig. 8) than the geometric algorithms. Control effort results at $x = 3$ should not be considered as some algorithms converged with as few as 2 or 3 time steps.

Analysing the results in which the initial *yaw* orientation is varied, it can be seen again that the control-oriented algorithms obtain better performance than the geometric ones. *3D-NLGL* and *3D-CC* perform similarly in terms of convergence time and path position. That is because, when the vehicle is close to the path, the distinct behaviour between these two algorithms on the approach to the path takes no remarkable effect. Also, it is important to note that the control-oriented algorithms make a significantly larger control effort to correct the *yaw* angle than the geometric algorithms. That is due to the aggressive *yaw* control that the *BS* and *FL* algorithms produce.

When the quadrotor deals with the effect of time-varying wind disturbances, represented in Fig. 14 and Table 7, *FL* algorithm performs worst. *BS* is the algorithm that handles best the external forces in terms of path distance error. Note that, apparently, the *FL* does not have sufficient strength to cope with wind, evidenced by its low average speed and the short distance along the path ($\gamma = 2.72$) that it is able to cover, compared with the other algorithms. The yaw error of geometric algorithms is larger than the one of *BS*, while the path distance error is similar.

Table 8 presents a comparison between the four algorithms summarizing the simulation result analysis and other qualitative indicators. This table sorts from best (1) to worst (4) the four algorithms on each qualitative aspect.

Regarding the design & tuning effort, the geometric algorithms (*3D-NLGL* and *3D-CC*) are easier to implement and they require only one parameter to tune. In contrast, the implementation of the control-oriented algorithms (*BS* and *FL*) is tedious since both present several complex derivatives and they have more parameters to tune. Furthermore, in the *FL* algorithm it is necessary to define a mathematical function (h_3) that calculates the minimum distance point on the path ($\gamma_{d_{\min}}$) given the position of the vehicle. This function depends exclusively on the geometry of the path and no solution is guaranteed to exist for every path.

Table 8 Qualitative comparison of the path following algorithms

	BS	FL	NLGL	CC
Path distance control	1	2	4	4
Yaw control	1	4	2	2
Velocity control	4	3	1	1
Convergence to the path	1	2	4	3
Computational effort	4	3	1	1
Wind disturbance	1	4	2	2
Control effort	3	4	2	1
Design & tuning effort	3	4	1	1
Path adaptability	3	4	1	1
State information requirements	3	3	1	1
Model-Based	3	3	1	1
Domain of attraction	1	4	2	2

The path adaptability defines the capability of each algorithm to adapt to different types of paths. The requirement of finding h_3 makes *FL* the worst for adapting to new path shapes. The *BS* algorithm is rated third since the defined path ($\mathbf{p}_d(\gamma)$) needs to be continuous and derivable. The geometric algorithms are again the best rated because they can be adapted to any path by changing only their specific control parameter.

The control-oriented algorithms require the full state information, while the geometric ones (along with the *PID*s of the autopilot) only require the position, attitude and the velocities on the x and y axis. Furthermore, the design of the control-oriented algorithms is model-based, while it is model-free in the geometric algorithms. Because of this, they can be applied to different kinds of vehicles. Both qualitative aspects are reflected in Table 8.

To end up, the domain of attraction of each algorithm is evaluated. The behaviour of the algorithms with the vehicle away from the path and in different initial conditions is analysed. *Backstepping* is global since it is based on the non-linear model and it includes a saturation on the position error term that results also in a saturation on the control actions. The *BS* algorithm is rated first on this characteristic since when the vehicle is far from the path and independently of the initial condition it is always able to converge to the path. *Feedback Linearisation* should be a global algorithm as well, since it is based on a full-state non-linear dynamic inversion. Nevertheless, the simulation results show that it is not really global, since it becomes unstable in specific initial conditions. That is, opposite to the *BS* case, the approach velocity of the *FL* algorithm grows unbounded as the distance to the path increases. Moreover, it gets unstable when the vehicle is in position $x = 0, y = 0$ since the third output (h_3) is indeterminate. This is the reason why the transient experiments start on $x = 0.5m$ and not on $x = 0$. For these reasons, the *FL* algorithm is rated worst in the domain of attraction aspect. Regarding the geometric algorithms, the *NLGL* has a domain of attraction defined by the distance L. When the vehicle is further away a special procedure must be performed, which may not assure stability. The *CC* is considered global, as the convergence to the path is assured for any position of the vehicle. Note that the analysis of the domain of attraction for the geometric algorithms depends exclusively on the position states, since

the rest of states (velocity, orientation and angular velocity) are controlled by the autopilot.

6 Conclusions

This paper deals with the trajectory control problem, which can be solved by path following or trajectory tracking. The concept of path following has been defined. According to the reviewed bibliography, it results in a smoother convergence to the path, less demand of control effort, a stronger robustness and other advantages over traditional trajectory tracking.

This survey paper gives a comprehensive review of PF algorithms. Several path following solutions applied to different types of UAVs, such as fixed-wing vehicles, helicopters and quadrotors, have been reviewed. Then, two control-oriented algorithms (*Backstepping* and *Feedback Linearisation*) and two geometric algorithms (*NLGL* and *Carrot-Chasing*) have been implemented to solve the path following problem on a quadrotor vehicle. Additionally, the geometric algorithms have been adapted to cope with three-dimensional paths. Finally, these four PF algorithms have been compared in simulation using a helix as a reference path.

Simulation results reveal that the *Backstepping* algorithm achieves the best performance in terms of path distance and yaw error as well as the best behaviour out of the path and at high velocities. However, it results in a very high computational effort and a significant control effort. Meanwhile, the *3D Carrot-Chasing* algorithm, in spite of its worse path distance performance, turns out to be easier to implement on any type of path, requires less state information and it results in a lower computational and control effort. Therefore, the authors consider that, among the implemented algorithms, these two are the best to solve the PF problem. The choice between them would depend on the problem requirements.

In this paper the path following algorithms have been compared in virtue of its intrinsic and theoretical characteristics. Other features related to the experimental platform, such as sensor models, uncertainties or communications, have been excluded from this survey. Drawing from the conclusions of this comparative study, future work is to apply the algorithms in real flight experiments. A previous step is to develop a realistic quadrotor path following simulation benchmark, in which the authors are already working.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

References

1. 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
2. Aguiar, A.P., Hespanha, J.P., Kokotovic, P.V.: Performance limitations in reference tracking and path following for nonlinear systems. Automatica **44**(3), 598–610 (2008). https://doi.org/10.1016/j.automatica.2007.06.030
3. Akhtar, A., Waslander, S.L., Nielsen, C.: Path following for a quadrotor using dynamic extension and transverse feedback linearization. In: 2012 IEEE 51st Annual Conference on Decision and Control (CDC), IEEE Conference on Decision and Control, pp. 3551–3556 (2012)
4. Akhtar, A., Waslander, S.L., Nielsen, C.: Fault Tolerant Path Following for a Quadrotor. In: 2013 IEEE 52nd Annual Conference on Decision and Control (CDC), Conference on Decision and Control, pp. 847–852 (2013)
5. Akkinapalli, V.S., Niermeyer, P., Lohmann, B., Holzapfel, F.: Adaptive nonlinear design plant uncertainty cancellation for a multirotor. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1102–1110 (2016). https://doi.org/10.1109/ICUAS.2016.7502555
6. Alexis, K., Papachristos, C., Siegwart, R., Tzes, A.: Robust model predictive flight control of unmanned rotorcrafts. J. Intell. Robot. Syst. **81**(3), 443–469 (2016). https://doi.org/10.1007/s10846-015-0238-7
7. Ali, S.U., Samar, R., Shah, M.Z., Bhatti, A.I., Munawar, K., Al-Sggaf, U.M.: Lateral guidance and control of UAVs using second-order sliding modes. Aerosp. Sci. Technol. **49**, 88–100 (2016). https://doi.org/10.1016/j.ast.2015.11.033
8. Ambrosino, G., Ariola, M., Ciniglio, U., Corraro, F., De Lellis, E., Pironti, A.: Path generation and tracking in 3-D for UAVs. IEEE Trans. Control Syst. Technol. **17**(4), 980–988 (2009). https://doi.org/10.1109/TCST.2009.2014359
9. Amidi, O., Thorpe, C.: Integrated mobile robot control. In: Mobile Robots V, vol. 1388 (1991). https://doi.org/10.1117/12.25494
10. Amin, R., Aijun, L., Shamshirband, S.: A review of quadrotor UAV: control methodologies and performance evaluation. Int. J. Autom. Control. **10**(2), 87–103 (2016). https://doi.org/10.1504/IJAAC.2016.076453
11. Baca, T., Loianno, G., Saska, M.: Embedded model predictive control of unmanned micro aerial vehicles. In: 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 992–997 (2016). https://doi.org/10.1109/MMAR.2016.7575273
12. Başçi, A., Can, K., Orman, K., Derdiyok, A.: Trajectory tracking control of a four rotor unmanned aerial vehicle based on continuous sliding mode controller. Elektronika Ir ElektrotechnikA **23**(3), 12–19 (2017)
13. Bouabdallah, S., Murrieri, P., Siegwart, R.: Design and Control of an Indoor Micro Quadrotor. In: 2004 IEEE International Conference on Robotics and Automation (ICRA), vol. 5, pp. 4393–4398 (2004). https://doi.org/10.1109/ROBOT.2004.1302409
14. Byrnes, C., Isidori, A.: Asymptotic stabilization of minimum phase nonlinear-systems. IEEE Trans. Autom. Control **36**(10), 1122–1137 (1991). https://doi.org/10.1109/9.90226
15. Cabecinhas, D., Cunha, R., Silvestre, C.: Rotorcraft path following control for extended flight envelope coverage. In: Proceedings of the 48th IEEE Conference on Decision and Control, 2009 Held Jointly with the 2009 28th Chinese Control Confer-

ence (CDC/CCC 2009), IEEE Conference on Decision and Control, pp. 3460–3465 (2009). https://doi.org/10.1109/CDC.2009.5400665
16. Cabecinhas, D., Cunha, R., Silvestre, C.: A globally stabilizing path following controller for rotorcraft with wind disturbance rejection. IEEE Trans. Control Syst. Technol. **23**(2), 708–714 (2015). https://doi.org/10.1109/TCST.2014.2326820
17. Camacho, E., Bordons, C.: Model Predictive Control. Advanced Textbooks in Control and Signal Processing. Springer, London (2004). https://books.google.es/books?id=Sc1H3f3E8CQC
18. Chen, H., Chang, K., Agate, C.S.: UAV path planning with tangent-plus-Lyapunov vector field guidance and obstacle avoidance. IEEE Trans. Aerosp. Electron. Syst. **49**(2), 840–856 (2013)
19. Chen, Y., Liang, J., Wang, C., Zhang, Y.: Combined of Lyapunov-stable and active disturbance rejection control for the path following of a small unmanned aerial vehicle. Int. J. Adv. Robot. Syst. **14**(2). https://doi.org/10.1177/1729881417699150 (2017)
20. Chen, Y., Liang, J., Wang, C., Zhang, Y., Wang, T., Xue, C.: Planar smooth path guidance law for a small unmanned aerial vehicle with parameter tuned by fuzzy logic. Journal of Control Science and Engineering **2017**, 11 (2017). https://doi.org/10.1155/2017/6712602
21. Chen, Y., Yu, J., Mei, Y., Wang, Y., Su, X.: Modified central force optimization (MCFO) algorithm for 3D UAV path planning. Neurocomputing **171**, 878–888 (2016). https://doi.org/10.1016/j.neucom.2015.07.044
22. Chen, Y., Yu, J., Su, X., Luo, G.: Path planning for multi-UAV formation. Journal of Intelligent & Robotic Systems **77**(1, SI), 229–246 (2015). https://doi.org/10.1007/s10846-014-0077-y
23. Cho, N., Kim, Y., Park, S.: Three-dimensional nonlinear differential geometric path-following guidance law. J. Guid. Control Dynam. **38**(12), 2366–2385 (2015). https://doi.org/10.2514/1.G001060
24. Choi, S., Kim, S., Jin Kim, H.: Inverse reinforcement learning control for trajectory tracking of a multirotor UAV. Int. J. Control. Autom. Syst. **15**(4), 1826–1834 (2017). https://doi.org/10.1007/s12555-015-0483-3
25. Cichella, V., Choe, R., Mehdi, S.B., Xargay, E., Hovakimyan, N., Kaminer, I., Dobrokhodov, V.: A 3d path-following approach for a multirotor uav on so(3). In: IFAC Proceedings Volumes, vol. 46, pp. 13–18 (2013). https://doi.org/10.3182/20131120-3-FR-4045.00039
26. Cichella, V., Kaminer, I., Dobrokhodov, V., Xargay, E., Choe, R., Hovakimyan, N., Aguiar, A.P., Pascoal, A.M.: Cooperative path following of multiple multirotors over time-varying networks. IEEE Trans. Autom. Sci. Eng. **12**(3), 945–957 (2015). https://doi.org/10.1109/TASE.2015.2406758
27. Dadkhah, N., Mettler, B.: Control system design and evaluation for robust autonomous rotorcraft guidance. Control. Eng. Pract. **21**(11), 1488–1506 (2013). https://doi.org/10.1016/j.conengprac.2013.04.011
28. Dubins, L.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Am. J. Math. **79**(3), 497–516 (1957). https://doi.org/10.2307/2372560
29. Edwards, C., Spurgeon, S.: Sliding Mode Control: Theory And Applications. Series in Systems and Control. Taylor & Francis, New York (1998). https://books.google.es/books?id=uH2RJhIPsiYC
30. Escareño, J., Salazar, S., Romero, H., Lozano, R.: Trajectory control of a quadrotor subject to 2d wind disturbances. Journal of Intelligent Robotic & Systems **70**(1), 51–63 (2013). https://doi.org/10.1007/s10846-012-9734-1
31. Faulwasser, T., Findeisen, R.: Nonlinear model predictive control for constrained output path following. IEEE Trans. Autom. Control **61**(4), 1026–1039 (2016). https://doi.org/10.1109/TAC.2015.2466911
32. Gandolfo, D.C., Salinas, L.R., Toibero, J.M., Brandao, A.: Path following for unmanned helicopter: an approach on energy autonomy improvement. Information Technology and Control **45**(1), 86–98 (2016). https://doi.org/10.5755/j01.itc.45.1.12413
33. García, C.E., Prett, D.M., Morari, M.: Model predictive control: Theory and practice-a survey. Automatica **25**(3), 335–348 (1989). https://doi.org/10.1016/0005-1098(89)90002-2
34. Gautam, A., Sujit, P.B., Saripalli, S.: Application of guidance laws to quadrotor landing. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 372–379 (2015)
35. Gerlach, A.R., Kingston, D., Walker, B.K.: UAV navigation using predictive vector field control. In: 2014 American Control Conference, pp. 4907–4912 (2014). https://doi.org/10.1109/ACC.2014.6859082
36. Gu, N., Wang, D., Liu, L., Zhang, B., Peng, Z.: Adaptive line-of-sight guidance law for synchronized path-following of under-actuated unmanned surface vehicles based on low-frequency learning. In: 2017 36th Chinese Control Conference (CCC), pp. 6632–6637 (2017). https://doi.org/10.23919/ChiCC.2017.8028408
37. Hamada, Y., Tsukamoto, T., Ishimoto, S.: Receding horizon guidance of a small unmanned aerial vehicle for planar reference path following. Aerosp. Sci. Technol. **77**, 129–137 (2018). https://doi.org/10.1016/j.ast.2018.02.039
38. Hartman, D., Landis, K., Mehrer, M., Moreno, S., Kim, J.: Quadcopter dynamic modeling and simulation (Quad-Sim). https://github.com/dch33/Quad-Sim (2014)
39. Heng, X., Cabecinhas, D., Cunha, R., Silvestre, C., Qingsong, X.: A trajectory tracking Lqr controller for a Quadrotor: design and experimental evaluation. In: TENCON 2015 - 2015 IEEE Region 10 Conference, pp. 1–7. https://doi.org/10.1109/TENCON.2015.7372729 (2015)
40. Jung, D., Ratti, J., Tsiotras, P.: Real-time implementation and validation of a new hierarchical path planning scheme of UAVs via hardware-in-the-loop simulation. J. Intell. Robot. Syst. **54**(1-3, SI), 163–181 (2009)
41. Jung, W., Lim, S., Lee, D., Bang, H.: Unmanned aircraft vector field path following with arrival angle control. Journal of Intelligent Robotic & Systems **84**(1), 311–325 (2016). https://doi.org/10.1007/s10846-016-0332-5
42. Kaminer, I., Yakimenko, O., Pascoal, A., Ghabcheloo, R.: Path generation, path following and coordinated control for time-critical missions of multiple UAVs. In: 2006 American Control Conference, Proceedings of the American Control Conference, vol. 1–12, p. 4906+ (2006). https://doi.org/10.1109/ACC.2006.1657498
43. Klausen, K., Fossen, T.I., Johansen, T.A., Aguiar, A.P.: Cooperative path-following for multirotor UAVs with a suspended payload. In: 2015 IEEE Conference on Control and Applications (CCA 2015), pp. 1354–1360 (2015)
44. Kokotovic, P.V., Sussmann, H.J.: A positive real condition for global stabilization of nonlinear systems. Syst. Control Lett. **13**(2), 125–133 (1989)
45. Kothari, M., Postlethwaite, I., Gu, D.W.: A suboptimal path planning algorithm using rapidly-exploring random trees. International Journal of Aerospace Innovations **2**, 93–104 (2009)
46. Kukreti, S., Kumar, M., Cohen, K.: Genetically tuned Lqr based path following for UAVs under wind disturbance. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 267–274. https://doi.org/10.1109/ICUAS.2016.7502620 (2016)

47. 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). https://doi.org/10.1007/s12555-015-0289-3
48. Li, L., Sun, L., Jin, J.: Survey of advances in control algorithms of quadrotor unmanned aerial vehicle. In: 2015 IEEE 16th International Conference on Communication Technology (ICCT), pp. 107–111 (2015)
49. Liang, Y., Jia, Y.: Combined vector field approach for 2D and 3D arbitrary twice differentiable curved path following with constrained UAVs. Journal of Intelligent Robotic & Systems **83**(1), 133–160 (2016). https://doi.org/10.1007/s10846-015-0308-x
50. Liu, P., Chen, A.Y., Huang, Y.N., Han, J.Y., Lai, J.S., Kang, S.C., Wu, T.H., Wen, M.C., Tsai, M.H.: A review of rotorcraft unmanned aerial vehicle (UAV) developments and applications in civil engineering. Smart Struct. Syst. **13**(6), 1065–1094 (2014)
51. Lou, W., Guo, X.: Adaptive trajectory tracking control using reinforcement learning for quadrotor. Int. J. Adv. Robot. Syst. **13**(1), 38 (2016). https://doi.org/10.5772/62128
52. Mahony, R., Kumar, V., Corke, P.: Multirotor aerial vehicles: modeling, estimation, and control of quadrotor. IEEE Robot. Autom. Mag. **19**(3), 20–32 (2012). https://doi.org/10.1109/MRA.2012.2206474
53. Manjunath, A., Mehrok, P., Sharma, R., Ratnoo, A.: Application of virtual target based guidance laws to path following of a quadrotor UAV. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 252–260. https://doi.org/10.1109/ICUAS.2016.7502565 (2016)
54. de Marina, H.G., Kapitanyuk, Y.A., Bronz, M., Hattenberger, G., Cao, M.: Guidance algorithm for smooth trajectory tracking of a fixed wing UAV flying in wind flows. In: 2017 IEEE International Conference on Tobotics and Automation (ICRA), pp. 5740–5745 (2017)
55. Martinsen, A.B., Lekkas, A.M.: Curved path following with deep reinforcement learning: Results from three vessel models. OCEANS 2018 MTS/IEEE Charleston, pp. 1–8 (2018)
56. Mayne, D., Rawlings, J., Rao, C., Scokaert, P.: Constrained model predictive control: Stability and optimality. Automatica **36**(6), 789–814 (2000). https://doi.org/10.1016/S0005-1098(99)00214-9
57. Mendoza-Soto, J.L., Corona-Sanchez, J.J., Rodriguez-Cortes, H.: Quadcopter path following control. a maneuvering approach. Journal of Intelligent & Robotic Systems. https://doi.org/10.1007/s10846-018-0801-0 (2018)
58. Miao, C.X., Fang, J.C.: An adaptive three-dimensional nonlinear path following method for a fix-wing micro aerial vehicle. Int. J. Adv. Robot. Syst. **9**, 1 (2012)
59. Micaelli, A., Samson, C., Robotique, P., Icare, P.: Trajectory tracking for unicycle-type and two-steering-wheels mobile robots. In: IFAC Proceedings Volumes, vol. 27, pp. 249–256 (1994)
60. Nelson, D.R., Barber, D.B., McLain, T.W., Beard, R.W.: Vector field path following for miniature air vehicles. IEEE Transactions on Robotics **23**(3), 519–529 (2007). https://doi.org/10.1109/TRO.2007.898976
61. Nie, C., Zheng, Z., Cai, Z.: Three-dimensional path-following control of a robotic airship with reinforcement learning. In: 2019 International Journal of Aerospace Engineering (2019)
62. Niermeyer, P., Akkinapalli, V.S., Pak, M., Holzapfel, F., Lohmann, B.: Geometric path following control for multirotor vehicles using nonlinear model predictive control and 3D spline paths. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 126–134. https://doi.org/10.1109/ICUAS.2016.7502541 (2016)
63. Ollero, A., Heredia, G.: Stability analysis of mobile robot path tracking. In: Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Interaction and Cooperative Robots, vol. 3, pp. 461–466 (1995). https://doi.org/10.1109/IROS.1995.525925
64. Ozbek, N.S., Onkol, M., Efe, M.O.: Feedback control strategies for quadrotor-type aerial robots: a survey. Trans. Inst. Meas. Control. **38**(5, SI), 529–554 (2016). https://doi.org/10.1177/0142331215608427
65. Park, S., Deyst, J., How, J.: Performance and lyapunov stability of a nonlinear path-following guidance method. J. Guid. Control Dynam. **30**, 1718–1728 (2007)
66. Raffo, G.V., Ortega, M.G., Rubio, F.R.: Backstepping/nonlinear h-infinity control for path tracking of a quadrotor unmanned aerial vehicle. In: 2008 American Control Conference, vol. 1–12, pp. 3356–3361 (2008)
67. Ratnoo, A., Hayoun, S., Granot, A., Shima, T.: Path following using trajectory shaping guidance. J. Guid. Control Dynam. **38**, 106–116 (2015). https://doi.org/10.2514/1.G000300
68. Roza, A., Maggiore, M.: Path following controller for a quadrotor helicopter. In: 2012 American Control Conference (ACC), pp. 4655–4660 (2012)
69. Rucco, A., Aguiar, A.P., Hauser, J.: Trajectory optimization for constrained UAVs: a virtual target vehicle approach. In: 2015 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 236–245. https://doi.org/10.1109/ICUAS.2015.7152296 (2015)
70. Rucco, A., Aguiar, A.P., Pereira, F.L., de Sousa, J.B.: A Predictive Path-Following Approach for Fixed-Wing Unmanned Aerial Vehicles in Presence of Wind Disturbances, vol. 427, pp. 623–634. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-27146-0_48
71. Rysdyk, R.: UAV path following for target observation in wind. J. Guid. Control Dynam. **29**(5), 1092–1100 (2006). https://doi.org/10.2514/1.19101
72. Shah, M.Z., Samar, R., Bhatti, A.I.: Lateral track control of UAVs using the sliding mode approach: from design to flight testing. Trans. Inst. Meas. Control. **37**(4), 457–474 (2015). https://doi.org/10.1177/0142331214543093
73. Shen, H., Guo, C.: Path-following control of underactuated ships using actor-critic reinforcement learning with Mlp neural networks. In: 2016 Sixth International Conference on Information Science and Technology (ICIST), pp. 317–321. https://doi.org/10.1109/ICIST.2016.7483431 (2016)
74. Singh, S., Padhi, R.: Automatic path planning and control design for autonomous landing of UAVs using dynamic inversion. In: 2009 American Control Conference, Vols 1–9, Proceedings of the American Control Conference, pp. 2409–2414 (2009)
75. Sontag, E.D., Sussmann, H.J.: Further comments on the stabilizability of the angular velocity of a rigid body. Syst. Control Lett. **12**(3), 213–217 (1989)
76. Stevšić, S., Nägeli, T., Alonso-Mora, J., Hilliges, O.: Sample efficient learning of path following and obstacle avoidance behavior for quadrotors. IEEE Robotics and Automation Letters **3**(4), 3852–3859 (2018). https://doi.org/10.1109/LRA.2018.2856922
77. Su, S.: Path following control of non-minimum phase VTOL aircraft via minimum distance projection method. In: 26th Chinese Control and Decision Conference (2014 CCDC), Chinese Control and Decision Conference, pp. 708–712 (2014)
78. Sujit, P.B., Saripalli, S., Sousa, J.B.: Unmanned aerial vehicle path following: a survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. IEEE Control Systems **34**(1), 42–59 (2014). https://doi.org/10.1109/MCS.2013.2287568
79. Valencia, D.R., Kim, D.: Trajectory tracking control for multiple quadrotors based on a neurobiological-inspired system. 2019

Third IEEE International Conference on Robotic Computing (IRC) pp. 465–470 (2019)
80. Van Loock, W., Pipeleers, G., Diehl, M., De Schutter, J., Swevers, J.: Optimal path following for differentially flat robotic systems through a geometric problem formulation. IEEE Trans. Robot. **30**(4), 980–985 (2014). https://doi.org/10.1109/TRO.2014.2305493
81. Wang, C., Song, B., Huang, P., Tang, C.: Trajectory tracking control for quadrotor robot subject to payload variation and wind gust disturbance. Journal of Intelligent Robotic & Systems **83**(2), 315–333 (2016). https://doi.org/10.1007/s10846-016-0333-4
82. Wang, T., Chen, Y., Liang, J., Wang, C., Zhang, Y.: Combined of vector field and linear quadratic Gaussian for the path following of a small unmanned helicopter. IET Control Theory Appl. **6**(17), 2696–2703 (2012). https://doi.org/10.1049/iet-cta.2012.0270
83. Wang, Y., Wang, X., Zhao, S., Shen, L.: Vector field based sliding mode control of curved path following for miniature unmanned aerial vehicles in winds. J. Syst. Sci. Complex. **31**(1), 302–324 (2018). https://doi.org/10.1007/s11424-018-8006-y
84. Yamasaki, T., Balakrishnan, S.N., Takano, H.: Integrated guidance and autopilot for a path-following UAV via high-order sliding modes. In: 2012 American Control Conference (ACC), Proceedings of the American Control Conference, pp. 143–148 (2012)
85. Zhao, B., Xian, B., Zhang, Y., Zhang, X.: Nonlinear robust sliding mode control of a quadrotor unmanned aerial vehicle based on immersion and invariance method. Int. J. Robust Nonlinear Control **25**(18), 3714–3731 (2015). https://doi.org/10.1002/rnc.3290
86. Zhaowei, M., Tianjiang, H., Lincheng, S., Weiwei, K., Boxin, Z., Kaidi, Y.: An iterative learning controller for quadrotor UAV path following at a constant altitude. In: 2015 34th Chinese Control Conference (CCC), pp. 4406–4411. https://doi.org/10.1109/ChiCC.2015.7260322 (2015)
87. Zhou, B., Satyavada, H., Baldi, S.: Adaptive path following for unmanned aerial vehicles in time-varying unknown wind environments. In: 2017 American Control Conference (ACC), pp. 1127–1132. https://doi.org/10.23919/ACC.2017.7963104 (2017)
88. Zhou, D., Schwager, M.: Vector field following for quadrotors using differential flatness. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 6567–6572 (2014)

Bartomeu Rubí received the M.S. degree in Automatic Systems and Industrial Electronics Engineering from the Universitat Politècnica de Catalunya (UPC) in 2015. Before that he received the B.S. degree in Industrial Electronics Engineering from the Universitat de les Illes Balears in 2013. Currently, he is working towards the PhD degree at the Research Center for Supervision, Safety and Automatic Control, UPC. The main research topic is focused on automatic control and machine learning applied on UAV.

Ramon Pérez degree and PhD in Physical Sciences by the Universitat de Barcelona (UB) in 1993 and Universitat Politècnica de Catalunya (UPC) in 2003 respectively. He has been lecturing in UPC since 1994. He has long experience in teaching modelling and control of dynamic systems. His research has been focussed on modelling, simulating, control and supervising water systems and UAV.

Bernardo Morcego is an Associate Professor at the Universitat Politècnica de Catalunya (UPC). He received a PhD degree in Computer Science from the UPC in 2000. He has been teaching several subjects in the area of automatic control in the schools of Engineering and Aeronautics in Terrassa and Barcelona. He is a member of the Research Center for Supervision, Safety and Automatic Control of UPC. His research interests include UAV control systems and computer vision applications.