Real-Time Path Generation and Obstacle Avoidance for Multirotors: A Novel Approach

Article *in* Journal of Intelligent & Robotic Systems · January 2018

DOI: 10.1007/s10846-017-0478-9

Phuong D. H. Nguyen
University of Hamburg
23 PUBLICATIONS 227 CITATIONS
SEE PROFILE

Carmine Tommaso Recchiuto
Università degli Studi di Genova
74 PUBLICATIONS 539 CITATIONS
SEE PROFILE

Antonio Sgorbissa
Università degli Studi di Genova
202 PUBLICATIONS 2,001 CITATIONS
SEE PROFILE

J Intell Robot Syst
DOI 10.1007/s10846-017-0478-9

Real-Time Path Generation and Obstacle Avoidance for Multirotors: A Novel Approach

Phuong D. H. Nguyen · Carmine T. Recchiuto · Antonio Sgorbissa

Received: 14 June 2016 / Accepted: 8 January 2017

Abstract Multirotors, among all aerial vehicles, are fundamental instruments in many situations, i.e. video recording of sport events, leisure, environmental monitoring before or after a disaster. In particular, in the context of environmental monitoring, the possibility of following a predetermined path while avoiding obstacles is extremely relevant. In this work, we propose a novel method for path definition in presence of obstacles, which describes a curve as the intersection of two surfaces. The planner, based on that path definition along with a Cascaded control architecture and utilizing a nonlinear control technique for both control loops (position and attitude), creates a framework to manipulate the multicopters' behaviors. The method is demonstrated to be able to generate a safe path taking into account obstacles perceived in real-time and avoids collisions. These algorithms are embedded in a software package to control the flight of a fully autonomous AscTec Firefly hexacopter with two cameras and onboard processing capabilities.

Keywords UAVs · Path following · Obstacle avoidance

P. D. H. Nguyen · C. T. Recchiuto (✉) · A. Sgorbissa
Universitá di Genova, DIBRIS, via all'Opera Pia 13, 16145 Genova, Italy
e-mail: carmine.recchiuto@dibris.unige.it

Mathematics Subject Classification (2010) 93C85 · 68T40

1 Introduction

The maneuverability of Unmanned Aerial Vehicles (UAVs), and specifically multirotors, is one of the principal reasons for the great attention that they are receiving from robotic researchers and the market. Indeed multirotors are endowed with a number of peculiarities that makes them extremely interesting in many scenarios: the ability of vertically taking-off and landing, ease of construction, low cost, flexibility, possibility to carry a payload (usually cameras or other sensors). The contexts of their applications include agriculture and vegetation mapping, monitoring of wetland ecosystems and coastal management, traffic monitoring, recording of sport events and damage assessment after a disaster (earthquakes, floods, avalanches, fires).

At the current state of the art, UAVs have been already used in real disaster scenarios, demonstrating their benefits. Some first examples were the utilization of two UAVs for the surveying of the damaged area after the Hurricane Katrina [1], and the Hurricane Wilma [2], and UAVs were efficiently deployed also in the context of the L'Aquila (Italy) heartquake [3], or more recently in the Lushan (China) heartquake [4]. In all these situations, aerial-based disaster assessment was an effective method for rescue teams, and disaster regions were efficiently and accurately identified, helping human operators to find survivors and target locations.

In the context of disaster scenarios, multirotors are required to fly in complex environments, in which buildings are likely to be completely or partially collapsed. Therefore, pre-existing maps are not very useful: debris, damaged furnitures and various objects are likely to occupy pre-existing free space, while new free spaces can be produced as well. Moreover, multirotors are usually teleoperated by a human operator, but some issues arise when a single human operator should deal with many robots at the same time: in particular when speaking of aerial vehicles, the equilibrium between the ease of use, the operator workload and the resulting situational awareness is extremely difficulty to be found [5]. For this reason, in recent applications, UAVs are often required to explore the environment autonomously [4, 6].

Summarizing, giving the presence of an unpredictable environment and the necessity of implementing autonomous strategies for monitoring, UAVs must have the ability to accurately follow a pre-defined exploration path, while being able to modify and update such paths in real-time in relation on how they perceive the environment. These tasks involves specific approaches for motion control, path following and obstacle avoidance.

This article starts with the analysis of current approaches for motion control and obstacle avoidance (see Section 2), and then presents the contribution of the current work, that is the definition of a novel algorithm for path generation and obstacle avoidance, using a novel technique to describe and adapt the path in presence of unexpected obstacles which requires very low computations and is therefore suitable for real-time operations. This planner is built with a controller based on the cascaded architecture and a nonlinear control technique, the feedback linearization, which is simple but effective. The description of the motion control strategy and of the approach for path generation with obstacle avoidance is given in Sections 3 and 4. All proposed algorithms for planning and control have been also implemented on real robots using only the available onboard sensors. The analysis of the performed experiments and a discussion of the results can be found in Section 5. Finally, Section 6 presents conclusions.

2 Related Work

As discussed in the Introduction, the task to be accomplished, i.e. path following in environments with obstacles, involves strategies for motion control, motion planning and obstacle avoidance.

In term of multirotor control, it is worth recalling some of the many control approaches proposed in the literature (mainly for quadrotors). Concerning linearization-based control techniques, the recent work of Ramírez et al. [7] proposing *PID control* techniques, and Bouabdallah [8], with both PID and LQR techniques deserve to be mentioned. While [7] presents a hierarchical control, where the fast dynamics is stabilized by using linearizing controllers based on classical PD methods, [8] proposed both a PD controller introduced for each orientation angle, and a *LQR approach*, where the system is linearized around each state, and the Sage-Eisenberg method is applied to solve the Riccati's equation.

Non-linear control laws have also been studied and exploited for multirotors. Among the others, Altuğ et al. [9] implemented *Exact input-output linearization* and a *Back-stepping controller* based on a small angle assumption on the yaw, applied to a real robot equipped with a camera and off-board image processing capabilities. *Exact linearization* and *Non-interacting control* have been described in [10]. The approach, tested only in simulation, involves a dynamic feedback controller which renders the closed-loop system linear, controllable and noninteractive after a change of coordinates in the state-space. Xu and Özgüner [11] proposed a *Sliding mode* control for quadrotors, providing a solution for fully actuated sub-systems and under-actuated sub-systems (using a cascaded structure), once again tested only in simulation. *Lyapunov-based, Back-stepping, Sliding mode* control and *Integral Back-stepping* techniques with cascaded structure have been implemented by [8], and validated both in simulation and real experiments.

The *Model Predictive Control* (MPC) has been widely used in this context: Bouffard et al. [12] described an approach based on a Learning-based

MPC that combines statistical learning with vehicle state estimation, Alexis et al. [13] proposed a switching MPC according to the linearized models of the quadrotor, and Bangura and Mahony[14] presented a dynamic reduction of the attitude dynamics along with feedback linearisation used together with MPC.

Finally, other control techniques are worth mentioning: a dynamic model of the rotorcraft obtained via a Lagrange approach and a controller based on Lyapunov analysis using a *Nested saturation algorithm* was implemented by Castillo et al. [15]; the concept of *Model Reference Adaptive Control* is used in combination with a nonlinear control structure based on the method of nonlinear, dynamic inversion in the work of Achtelik et al. [16]; a *Dynamic inversion-based* two-loop controller (with position as outer loop and rate as inner loop) has been presented in [17]. Table 1 summarizes these principal approaches for motion control with multirotors.

Unlike multicopter control, path following and obstacle avoidance have not been paid a great attention by the research community in the multirotors domain.

Concerning path following, almost all control approaches make use of the concept of Virtual Target Point (VTP), i.e. a target moving in space, used as reference point for the convergence to the path. Methods involving the usage of VTPs are for instance *Charrot Chasing* approaches or *Non-linear Control Laws (NLGL)*. In the first approach, implemented in the works of Kim and Shin [18] and Tabatabaei et al. [19], the VTP position (figuratively, the charrot) is updated on-line in relation to the path to be followed and the UAV (figuratively, the horse chasing the carrot) moves along consecutive waypoints. In the NLGL approach the VTP moves in a more complex fashion and the stability of the guidance law is guaranteed using Lyapunov stability arguments (examples can be found in the works of Aguiar and Hespanha [20] and Frazzoli et al. [21]). Other techniques involve *Vector Fields*, where the vehicle is driven with the aid of a vector field that generates, for each point in the space, the target direction for the convergence of the UAV to the path (also in this case the convergence can be ensured using VTP [22]) and *Pure Pursuit and Line-of-Sight (PLOS)* based path following, that takes inspiration from the method for controlling missiles, combining a Pure Pursuit control law, that moves the UAV toward the next waypoint, and the LOS approach, that controls the best angle in relation to the adjacent waypoints [23].

Finally, other methods are based on the implementation of *LQR controllers*, as in the work of

Table 1 Motion control approaches for UAVs

Motion Control Approach	Implementation
PID controller	Ramírez et al. [7], Bouabdallah [8]
LQR controller	Boubdallah [8]
Exact input-output linearization, Back-stepping	Altuğ et al. [9]
Exact linearization, Noninteracting control	Mistler et al. [10]
Sliding Mode	Xu and Özgüner [11], Bouabdallah [8]
Lyapunov based, Integral Back-stepping	Boubdallah [8]
Model Predictive Control	Bouffard et al. [12], Alexis et al. [13], Bangura and Mahony [14]
Nested Saturation	Castillo et al. [15]
Adaptive Control	Achtelik et al. [16]
Dynamic Inversion Based	Achtelik et al. [17]

Osborne and Rysdyk [24], and on *model-based predictive controllers* (MPC), where the control law, based on the modeling of the vehicle, sets to zero the error between the tern of the vehicle and the tern of a virtual reference [25]. Table 2 summarizes all the above approaches for path following.

Obstacle avoidance techniques constitute also a relatively recent field in this research area; nonetheless there are some impressive milestones in the field. Most of the successful classical approaches rely on the concept of *Artificial Potential Fields*: the method is based on computing artificial repulsive forces, which are exerted on the robot by surrounding obstacles, and an attractive force exerted by the goal. The forces are summed up to produce a resulting force vector, used to control the motion of the robot. Although the approach has well known problems when dealing with robot navigation, among which the presence of local minima in the field, it has been widely used also when dealing with UAVs [26, 27].

More recent approaches explored different methods, such as the *Rapidly exploring Random Tree* (RRT), that grows a tree rooted at the start configuration by using random samples from the search space and creates edges between the samples if the connection is feasible (passes entirely through free space). Achtelik et al. [28] applied a modified implementation of the algorithm, the Rapidly Exploring Random Belief Tree (RRBT) algorithm, to plan a collision-free path considering the uncertainty in state estimation and the quadcopter's controller dynamics, while Kinodymanic RRT*, an optimized version of RRT that minimizes the distance from the root to the tree nodes in each iteration, was applied to quadrotors in [29]. The *Belief Roadmap*, an information-space extension of the Probabilistic Roadmap, along with the Unscented Kalman Filter for planning motion trajectory of quadcopter in GPS-denied environment and avoiding obstacles, has been presented in [30]. The *Minimum Snap Trajectory Generation* algorithm has been described in [31] and it allowed a UAV to safely pass through specific corridors, satisfying constraints on velocities and accelerations. The method is based on the generation of optimal trajectories in the sense that they minimize cost functionals derived from the square of the norm of the snap (the fourth derivative of the position). Finally it is also worth of mention the implementation of the *Dynamic Window Approach* [32]: the approach consists in a real-time collision avoidance strategy for mobile robots that is directly derived from the dynamics of the robot, dealing with the constraints imposed by limited velocities and accelerations. Please refer to Table 3 for a summary of the most common obstacle avoidance approaches and their implementation with UAVs.

All these approaches for path following and obstacle avoidance deal with a prior knowledge of obstacle-free environment and have a complex implementation, or present well known problems such as the possibility to be stranded at points of local minima (APFs). Taking into account these problems, the proposed approach, while adopting a well-estabilished nonlinear control technique, the feedback linearization, presents a novel algorithm that merges path following and obstacle avoidance, can deal with changing environments, and relies on a simple representation of the path to be followed. The next Sections will describe the approach in details.

It must be here underlined that, while the presented approach can be implemented with any robot, it is particularly suited for underactuated robots. Indeed, the control of underactuated robots is quite complex since they are multi-input, non linear, nonminimum-phase systems: thus, the challenge consists in meeting the

Table 2 Path following methods for UAVs

Path Following Approach	Implementation
Charrot Chasing	Kim and Shim [18], Tabatabaei et al. [19]
NLGL	Aguiar and Hespanha [20], Frazzoli et al. [21]
PLOS	Kothari et al. [23]
Vector FIelds approach	Nelson et al. [22]
LQR	Osborne and Rysdyk [24]
MPC	Alexis et al. [25]

Table 3 Obstacle Avoidance strategies for UAVs

Obstacle Avoidance Approach	Implementation
Artificial Potential Fields	Rezaee and Abdollahi [26], Nieuwenhuisen et al. [27]
Rapidly exploring Random Tree	Achtelik et al. [28], Webb and van der Berg [29]
Probabilistic Roadmap	He et al. [30]
Minimum Snap Trajectory Generation	Mellinger and Kumar [31]
Dynamic Window Approach	Vista et al. [32]

desired specifications while ensuring internal stability [33]. Given these aspects, the utilization of a Dynamic state feedback approach guarantees internal stability when dealing with nonlinear systems [34].

3 Control Strategy

The mathematical model of the multicopter dynamics can be developed from Newton-Euler formalism [35] in the state space form:

$$\dot{X} = f(X, U) = \begin{Bmatrix} \dot{\phi} \\ \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1U_2 \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}a_3 - \dot{\phi}a_4\Omega_r + b_2U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}a_5 + b_3U_4 \\ \dot{z} \\ g - (\cos\phi\cos\theta)\frac{1}{m}U_1 \\ \dot{x} \\ U_x\frac{1}{m}U_1 \\ \dot{y} \\ U_y\frac{1}{m}U_1 \end{Bmatrix} \tag{1}$$

where the components of the state X and the control vector U (for hexacopters) are expressed as:

$$\begin{cases} X = [\phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi}, z, \dot{z}, x, \dot{x}, y, \dot{y}]^T \\ U = [U_1, U_2, U_3, U_4]^T, \end{cases} \tag{2}$$

with

$$\begin{cases} U_1 = b\sum_{i=1}^{6}\Omega_i^2 \\ U_2 = b\big(\Omega_1^2c + \Omega_2^2 + \Omega_3^2c - \Omega_4^2c - \Omega_5^2 - \Omega_6^2c\big) \\ U_3 = b\big(\Omega_1^2 - \Omega_3^2 - \Omega_4^2 + \Omega_6^2\big)s \\ U_4 = d\big(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2 + \Omega_5^2 - \Omega_6^2\big), \end{cases} \tag{3}$$

and the coefficients in Eq. 1, 3 are defined as:

$$\begin{cases} a_1 = \frac{I_y - I_z}{I_x} \quad a_3 = \frac{I_z - I_x}{I_y} \quad a_5 = \frac{I_x - I_y}{I_z} \\ a_2 = \frac{J_r}{I_x} \quad a_4 = \frac{J_r}{I_y} \\ b_1 = \frac{l}{I_x} \quad b_2 = \frac{l}{I_y} \quad b_3 = \frac{1}{I_z} \\ c : \cos 60^o \quad s : \sin 60^o \\ U_x = -(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ U_y = -(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \\ \Omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 - \Omega_5 + \Omega_6. \\ b\text{: thrust coefficient;} \quad d\text{: drag coefficient} \end{cases} \tag{4}$$

The Eq. 3 illustrates the mapping between the control signals and the speed of propellers, called mapping matrix, which deeply depends on the structure of the multicopter platform ("$\sin 60^o$" and "$\cos 60^o$") and is different for different types of multicopters. By introducing the control vector U, the dynamics model is separated from the physical properties of the multicopter (e.g. number of propellers), so that any following designed controller and planner, which are based on the proposed model, are comfortably applicable to any type of multicopter.

In order to fully control all outputs of the multicopter system, which is complex, non-linear and under-actuated, a Cascaded control architecture (Fig. 1) composed of two control loops (position as outer loop and attitude as inner loop) has been chosen.

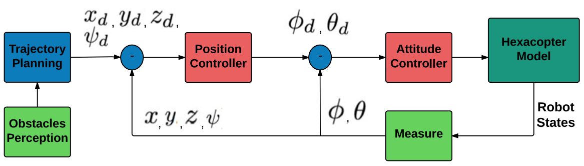


Fig. 1 Cascaded control architecture for position tracking. The red blocks show the controllers, including Attitude Controller in inner loop and Position Controller for outer one, while the green and blue ones display the measurement, planner and system parts

The main idea underlying the cascaded schema is to achieve a stable noninteraction by means of the outer control feedback. Given a nonlinear underactuated system, this approach is suitable for finding a feedback law such that internal stability is guaranteed [33]. One additional advantage of this architecture is that it can take benefit of a well-tuned attitude controller (i.e. the AutoPilot controller), which is embedded in most multicopters nowadays. By using this architecture, the proposed control law can also be applied to other robot platforms with very little modifications and tuning effort, even though they have different frame architectures (i.e., 4,6 or 8 propellers).

We can separate the system controller into 2 distinct ones because the mathematical model (Eq. 1) shows that the position dynamics depends on the attitude one but not vice versa. The Position Controller receive the errors of the robot's position (x, y, z) and heading (ψ) and their desired value (x_d, y_d, z_d, ψ_d) as input and calculate the desired value of roll and pitch angle (ϕ_d and θ_d) as output. These two values are compared with measured values ϕ and θ and then their errors are fed to the Attitude Controller to calculate controlled value for the robot model. Besides, the Position Controller computes controlled height and heading value for output as well.

It can be seen how the multicopters' model is highly non-linear and under-actuated, so linearization based control techniques (e.g. PID and LQR) require a linearization step for each hovering point (equilibrium point) and multicopters can only operate stably near hover and in absence of large disturbances. For path following, the robot should be able to change its position continuously and without changing modes of the controller or sets of tuned parameters. Therefore, a non-linear control techniques has been implemented: Dynamic state feedback [36], a simple but efficient non-linear control technique, has been chosen as the main control law (with only one set of control parameters) for every operation mode of the robot, including take-off, landing, position tracking and path following. The main idea of the Dynamic state feedback technique is the introduction of a virtual control to transform a non-linear system to a linear one without any approximation; then linear control theory can be used. Moreover, it is well known that, for systems with time-invariant uncertainties, the tolerable uncertainty bound may be improved by using a Dynamic state feedback controller compared to the static feedback one [37]. Indeed, it has been theoretically demonstrated that using a Dynamic state feedback approach guarantees that a higher robustness can be achieved with respect of linearization based control techniques [38]. LQR controllers and linear controllers in general have a very small margin with respect to the variations of the gain and/or phase of the open-loop plant. On the contrary, by using Dynamic state feedback and by properly choosing a stable transfer function, it is possible to obtain a controller that is robustly Hurwitz, i.e. the gain margin can be arbitrarily large [38].

From Eq. 1 the control U can be calculated as following:

$$\begin{cases} U_1 &= \frac{m}{cos\phi cos\theta}\left(g - V_1\right) \\ U_2 &= \frac{1}{b_1}\left(V_2 - \dot{\theta}\dot{\psi}a_1 - \dot{\theta}a_2\Omega_r\right) \\ U_3 &= \frac{1}{b_2}\left(V_3 - \dot{\phi}\dot{\psi}a_3 + \dot{\phi}a_4\Omega_r\right) \\ U_4 &= \frac{1}{b_3}\left(V_4 - \dot{\theta}\dot{\phi}a_5\right) \end{cases} \tag{5}$$

and

$$\begin{cases} U_x &= \frac{m}{U_1}V_x \\ U_y &= \frac{m}{U_1}V_y \end{cases} \tag{6}$$

with the virtual control inputs $V_1, V_2, V_3, V_4, V_x, V_y$ calculated using the PID tracking law as follows:

$$\begin{cases} V_1 &= \ddot{z_d} + k_{d1}\dot{e_z} + k_{p1}e_z + k_{i1}\sum e_z\delta T \\ V_2 &= \ddot{\phi_d} + k_{d2}\dot{e_\phi} + k_{p2}e_\phi + k_{i2}\sum e_\phi\delta T \\ V_3 &= \ddot{\theta_d} + k_{d3}\dot{e_\theta} + k_{p3}e_\theta + k_{i3}\sum e_\theta\delta T \\ V_4 &= \ddot{\psi_d} + k_{d4}\dot{e_\psi} + k_{p4}e_\psi + k_{i4}\sum e_\psi\delta T \\ V_x &= \ddot{x_d} + k_{dx}\dot{e_x} + k_{px}e_x + k_{ix}\sum e_x\delta T \\ V_y &= \ddot{y_d} + k_{dy}\dot{e_y} + k_{py}e_y + k_{iy}\sum e_y\delta T. \end{cases} \tag{7}$$

Note that the desired values of roll and pitch angles can be calculated as:

$$\begin{cases} \phi_d &= sin^{-1}(U_y\cos\psi - U_x\sin\psi) \\ \theta_d &= sin^{-1}\left(\frac{-U_x - \sin\phi\sin\psi}{\cos\phi\cos\psi}\right). \end{cases} \tag{8}$$

The convergence of the proposed control algorithm can be demonstrated. Starting from the dynamics of the roll angle (1):

$$\ddot{\phi} = \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1U_2 \tag{9}$$

The control signal U_2 can be calculated as:

$$U_2 = \frac{1}{b_1}\left(V_2 - \dot{\theta}\dot{\psi}a_1 - \dot{\theta}a_2\Omega_r\right) \tag{10}$$

with V_2 being a virtual (immediate) control, introduced to convert the dynamics of the roll angle into linear form as:

$$\ddot{\phi} = V_2 \tag{11}$$

Since we want to asymptotically stabilize the roll dynamics to the desired value ϕ_d, we can use the PID tracking law to formulate this control V_2 as below:

$$V_2 = \ddot{\phi}_d + k_{d2}(\dot{\phi}_d - \dot{\phi}) + k_{p2}(\phi_d - \phi) + k_{i2} \sum (\phi_d - \phi)\delta T \tag{12}$$

As seen in the Eq. 12, the error dynamics of the roll angle can be written as:

$$(\ddot{\phi}_d - \ddot{\phi}) + k_{d2}(\dot{\phi}_d - \dot{\phi}) + k_{p2}(\phi_d - \phi) + k_{i2} \sum (\phi_d - \phi)\delta T = 0 \tag{13}$$

or:

$$\ddot{e_\phi} + k_{d2}\dot{e_\phi} + k_{p2}e_\phi + k_{i2} \sum e_\phi \delta T = 0 \tag{14}$$

By tuning the control parameters k_{d2}, k_{p2} and k_{i2}, the behavior of the roll angle of the robot can be modified, making the roll dynamics to be asymptotically stable.

Equations 13 or 14 proves that the choice of the virtual control V_2 allows the roll dynamics to be controlled as desired. A similar procedure can be applied for other controlled variables, i.e. pitch, yaw, x, y, z.

4 Path Generation with Obstacle Avoidance

4.1 Path Definition

In this approach, the 3D path used for multicopter's motion is defined through the intersection of two surfaces, each one represented by an implicit equation in the form $f_i(x, y, z) = 0$, $i = 1, 2$. The path is given by the solutions of the system below:

$$\begin{cases} f_1(x, y, z) = 0 \\ f_2(x, y, z) = 0. \end{cases} \tag{15}$$

For example, two intersecting planes can be described using the equations $a_i x + b_i y + c_i z + d_i = 0$, $i = 1, 2$, by properly choosing the coefficients a_i, b_i, c_i, d_i. The surfaces should be chosen so that there are always valid solutions for the system (15). A counter example could be using two parallel planes, for which such solutions do not exist.

Some example of paths created with the above idea are expressed in Eq. 16 as a straight line or in Eq. 17 as an ellipse (the intersection of a cylinder with elliptical base and a flat surface), as following:

$$\begin{cases} f_1(x, y, z) = a_1 x + b_1 y + c_1 z + d_1 = 0 \\ f_2(x, y, z) = a_2 x + b_2 y + c_2 z + d_2 = 0 \end{cases} \tag{16}$$

and

$$\begin{cases} f_1(x, y, z) = (x - a_1)^2 + (y - b_1)^2 - c_1^2 = 0 \\ f_2(x, y, z) = a_2 x + b_2 y + c_2 z + d_2 = 0 \end{cases} \tag{17}$$

with all coefficients a_i, b_i, c_i, d_i chosen properly. Figure 2 shows the definition of an elliptic path as expressed in Eq. 17.

For the special case of a 2D path, e.g. the robot tries to maintain its flying height and moves on a 2D plane, it is easy to set one of the surfaces in Eq. 15 as describing a plane at a constant height (e.g., $f_2(x, y, z) = z = H$, with H being the desired height) and choose the other surface according to the path to be followed. Without loss of generality, it can be chosen the zero-height plane as the reference plane. In this case, the function describing the path turns out to contain only 2 variables, and it can be re-written as follows (using the implicit equation of a curve on the plane):

$$f(x, y) = 0. \tag{18}$$

As the results, the above line (Eq. 16) and curve (Eq. 17) can be described respectively as:

$$f = ax + by + c = 0 \tag{19}$$

and

$$f = (x - a)^2 + (y - b)^2 - c^2 = 0 \tag{20}$$

The path representation described in Eqs. 15 and 18, exhibits two main advantages for path following:

– First, when the robot is located in x, y, z, the value of the function $f_i(x, y, z)$, $i = 1, 2$ in such position provides hints to drive the robot towards the respective surface (and hence to the intersection of the two surfaces). In fact, $f_i(x, y, z) = 0$ only happens when the position x, y, z belongs to that surface. In other cases, the value of the function depends on the distance from the vehicle to the surface, i.e., the further is the robot, the higher is the absolute value. The sign of the returned value

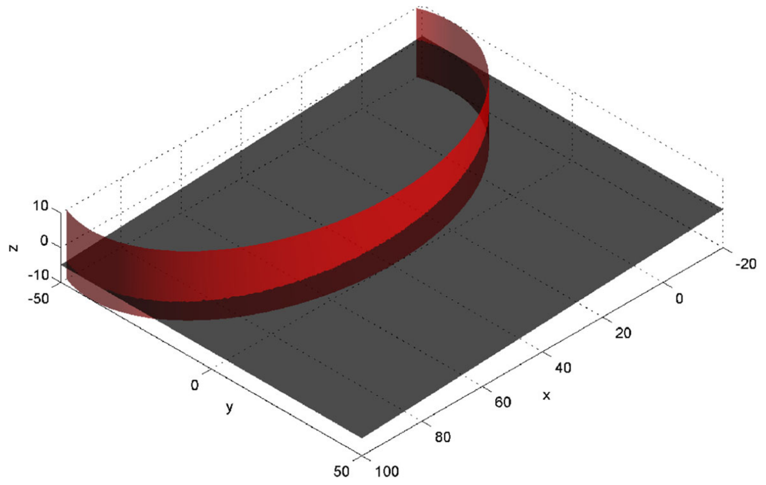


Fig. 2 Definition of an elliptic path through surface intersection

depends on which side (i.e., the two subspaces divided by the surface) the position belongs to.

– More importantly, a path defined in this form can be easily deformed in presence of obstacles, through an approach that requires a very low computation effort and is suitable for real-time (or on-the-flight) operations. The details will be clearly shown in the next section.

Moreover, details about this technique for path representation can be found in [39, 40].

4.2 Path Modification in Presence of Obstacles

In order to avoid perceived obstacles during the flight, the robot's path should be deformed when it perceives the presence of any obstacle by modifying the path description functions in Eq. 15.

Towards this end, each obstacle is modeled by an obstacle function $O_j(x, y, z)$, and the path is modified by adding the obstacle function to one or both surface functions in Eq. 15. In presence of more obstacles, it is necessary to sum up all individual obstacles' contributions.

In the following, we assume that only one surface, e.g. $f_1(x, y, z)$, is modified to consider the presence of obstacles. That is, the modified path is described as:

$$\begin{cases} f_1'(x, y, z) = f_1(x, y, z) + \sum_{j=1}^{N} O_j(x, y, z) = 0 \\ f_2(x, y, z) = 0. \end{cases} \tag{21}$$

Suppose that the two surfaces are defined as having the following properties:

– the gradient ∇f_1 of $f_1(x, y, z)$ is never parallel to the z axis of the world frame (a counter example would be a plane parallel to the ground plane); thus the surface $f_1(x, y, z) = 0$ mostly gives information about the desired direction of motion of the vehicle along the xy axes;

– The gradient ∇f_2 of $f_2(x, y, z)$ always has a non-null component along the z axis (a counter example would be a plane perpendicular to the ground plane); thus, $f_2(x, y, z) = 0$ mostly gives information about the desired altitude of the vehicle along the z axis.

With the above properties, operating on $f_1(x, y, z)$ means avoiding obstacles mostly along the xy direction (operating on $f_2(x, y, z)$ would have the effect of avoiding obstacles by changing the height).

The obstacle function $O_j(x, y, z)$ deserves attention. Since it must be chosen to keep the multicopter away from the position of obstacles (x_j, y_j, z_j), it must hold:

$$\begin{cases} |O_j(x, y, z)| \gg 0, & \text{if } \ |(x, y, z)^T - (x_j, y_j, z_j)^T| \approx 0 \\ |O_j(x, y, z)| \approx 0, & \text{if } \ |(x, y, z)^T - (x_j, y_j, z_j)^T| \gg 0. \end{cases} \tag{22}$$

One possible solution is using a Gaussian function described as:

$$O_j(x, y, z) = A_j e^{-\frac{(x-x_j)^2+(y-y_j)^2+(z-z_j)^2}{\sigma^2}} \tag{23}$$

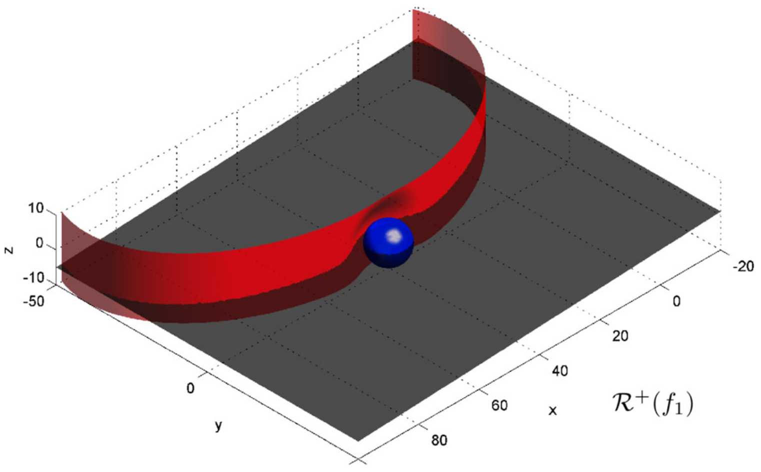


Fig. 3 Deformation of the path by the presence of an obstacle

with A_j (amplitude of the Guassian function) and σ (standard deviation of the Gaussian function) tuning parameters that must be properly chosen to shape the form of the modified surface $f_1'(x, y, z) = 0$ (which, in turn, change the shape of obstacle avoiding path).

Figure 3 shows these concepts: the path is obtained by intersecting a plane with an ellipsoid, which is deformed by the presence of an obstacle. The result is a path that avoids the obstacles, while staying as closer as possible to the original elliptic path.

It should be noticed that, by computing the tuning parameters A_j and σ properly, it is possible to guarantee that the deformed path does not collide with obstacles, even in the presence of multiple ones [41][1]. Besides, the path generated by this method is able to get rid of the local minima problem [41], which is the main drawback of Artificial Potential Fields and other local methods [42]. By ignoring details, the reason for the absence of local minima can be briefly motivated as follows. Assume that the goal and the start are connected by the original path defined by Eq. 15: since the appearance of obstacles can only deform the path (21) but it cannot disconnect it, the deformed path is guaranteed to drive the robot from start to goal by avoiding all obstacles.

Also, it is very important to outline that the obstacle detection and the path modification can be performed in real-time due to the fact that, above a given distance between the original path and the obstacles, the effect of the Gaussian contributions $O_j(x, y, z)$ is almost null. Hence, the calculation of the obstacle avoidance path only needs to consider the obstacles in a local map representing the surroundings of the robots, rather than in a global one.

The above mentioned properties of the algorithm have been demonstrated in [41].

Similarly to the previous Section, if we limit our analysis to motion on a plane, the 2D path in presence of obstacles can be re-written as follows:

$$f'(x, y) = f(x, y) + \sum_{j=1}^{N} O_j(x, y) = 0 \tag{24}$$

Equations 21 and 24 are the ones adopted in simulations and experiments in the following Sections.

4.3 Path Generation

As mentioned in Section 3, starting from a given path (21) it is necessary to generate desired poses as inputs to the position controller. Since the position controller requires input signals in sampled form (to be fed into it at every sampling period), the path described in the form of Eq. 21 is approximated as shown in Fig. 4.

Assume, in the following, that the robot is moving on a plane and hence the path is given by a single function $f(x, y) = 0$, which possibly takes into account the presence of obstacles as described in the previous Section. Also, suppose that the multicopter is flying at the position $P_k(x, y)$ at time instant T_k. The next

[1]How it will be shown later, while the method ensures that the path does not collide with obstacles, for small values of A and σ, considering the volume occupied by the robot and errors of the localization system, a collision between the robot and the obstacle can still occur. However, by appropriately increasing the value of the two parameters, this situation can be avoided.

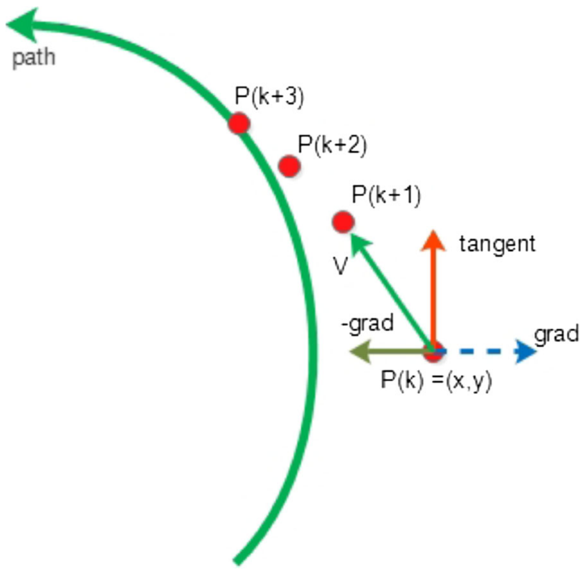


Fig. 4 Path generation by using gradient and tangent vector

desired position should be generated in the following form:

$$P_{k+1}(x, y) = P_k(x, y) + k_V V_k(x, y)\Delta T \tag{25}$$

where $V_k(x, y)$ is the "velocity vector" at time instant T_k and can be calculated as follows:

$$V_k(x, y) = -k_{grad} f(x, y)\nabla f(x, y) + t(x, y) \tag{26}$$

and

- $f(x, y)$ is the value of the function at the position $P_k(x, y)$, taken as a measure of the error between the ideal path and the real position;
- $\nabla f(x, y) = (f_x, f_y)$, is the gradient of f evaluated at the position $P_k(x, y)$ (the blue dashed arrow in Fig. 4); the product $-f(x, y)\nabla f(x, y)$ drives the robot towards the desired path;
- $t(x, y) = (-f_y, f_x)$ is the tangent to the level curve in (x, y), a vector perpendicular to the vector $\nabla f(x, y)$ in the same plane of $\nabla f(x, y)$ (the orange arrow in the Fig. 4). Notice that there are two different $t(x, y)$ with opposite directions, and they should be properly chosen to create the desired motion.

The two tuning gains in Eq. 25 affect the path following behavior of the multicopter as follows:

- k_V is referred to as the velocity vector weight, which determines how far the next position $P_{k+1}(x, y)$ is away from the current position $P_k(x, y)$ or, in other words, how fast is the movement of robot;
- k_{grad} is referred to as the gradient vector gain, which weighs the contribution of the gradient vector in computing the velocity vector (i.e., it pushes the robot back to the path faster or slower). A "too high" value of k_{grad} will produce an overshoot in the motion of the multicopter about the desired path.

In addition, notice that the value of the function $f(x, y)$ in Eq. 26 can be interpreted as an adaptive gain for the gradient vector. As already explained, the value of $f(x, y)$ depends on the distance between the current position and the desired path, and is equal to zero if and only if the robot is on the path (in this case, there is only the contribution of the tangent vector in the velocity vector).

In the more general case that the path does not lie on a plane, the computation of V_k is generalized by considering three weighted contributions: the two gradients ∇f_1 and ∇f_2, as well as the tangent to the path, i.e. $\nabla f_1 \times \nabla f_2$.

5 Simulation and Experiments

5.1 Experimental Setup

Tests have been performed both in simulation and with a real multirotor. For simulating, the mathematical functions of the multicopter's dynamic model, the controller and the path generator have been the basis to build a simulated model of the whole system, realized on Matlab/Simulink [43]. Moreover, further simulation tests have been carried out by using the robotic simulator Gazebo [44] and the RotorS MAV Framework [45], by applying the integrated path following and obstacle avoidance method to a simulated Firefly hexacopter (Fig. 5) from Ascending Technologies GmbH, Germany. Real experiments have been finally conducted with the real Firefly hexacopter.

The whole hardware architecture of the multicopter platform includes 2 processing boards, namely AutoPilot and Mastermind, and is embedded with a pre-built Ubuntu Linux (version 12.04-2) running on the Mastermind board. On the top of this Linux kernel, we use the ETHNOS environment [46, 47], a

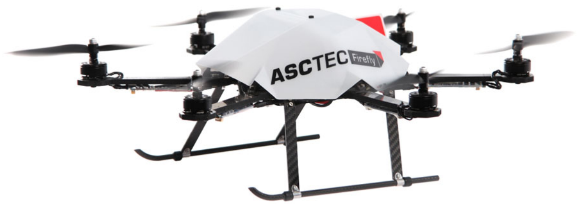


Fig. 5 Firefly hexacopter

dedicated distributed real-time Operating System, to handle all communication and execution tasks of the Firefly.

All computations are performed on–board only, using two on–board cameras for environmental perception (i.e., no external motion capture system is adopted). The localization is based on the artificial vision library Aruco [48]. The principal functionality of the library is to recognize up to 1024 different markers, applying an Adaptive Thresholding and the Otsu's algorithm. When a marker is recognized, the relative distance and orientation of the camera with respect of the marker is calculated.

Figure 6 shows the testing environment. In the current setup, an area of 4.2m x 4.2m has been considered for the experiments. The robot is constantly facing a wall with reference markers, that is used for localization, whereas the boxes with different markers represent obstacles. This simplification is proposed since, in the present work, we focus on control aspects rather than perception; whereas in a real-world implementation, more sophisticate solutions for localization and obstacle perception should be adopted. As the reader can imagine, the localization system is affected by a certain noise; in particular, the farther the distance from the markers, the higher are the errors in measurement. This effect contributes to the difference that can be observed between the simulated behaviour of the multicopter and the real one (see the next sections).

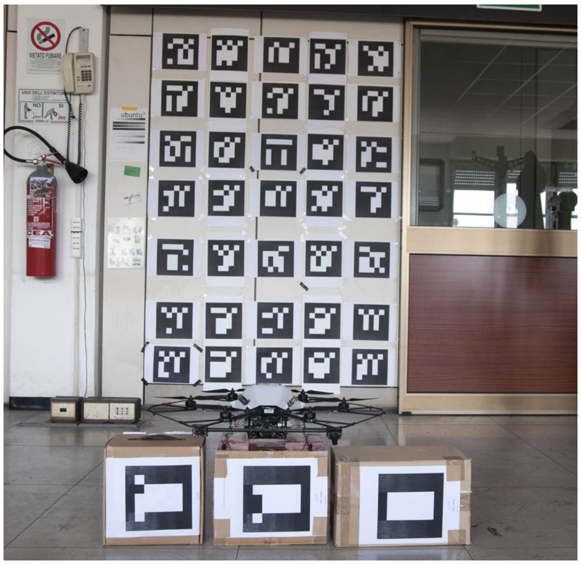


Fig. 6 Experimental environment

As a matter of fact, the uncertainties in pose estimation using ArUco library have the effect of producing many *false* obstacles, even in the case that both the robot and obstacle do not move. In particular, even a very small error in the robot's pose leads to the detection of another obstacle. The same thing also happens as a consequence of perceptual errors when computing the obstacle's pose w.r.t. the camera. In order to reduce this oscillation of obstacle's pose and/or to prevent the creation of new *false* obstacles, the current approach adapts the Occupancy Grid Map method, originally used for mapping as in [42]. The main idea of this method is creating a map of grid cells with a certain dimension (e.g. 0.1 x 0.1 x 0.1 m) and storing in each grid cell the number of times that the obstacle has been detected in the corresponding cell. The higher the number in a grid cell, the higher is the probability that an obstacle exists in the real world in the corresponding position; therefore a threshold has been set for considering a cell as occupied by an obstacle. The optimal value of this threshold has not been deeply investigated, since the detection of the obstacle and the analysis of the environment was not the focus of this work.

Finally, it should be outlined that the flying area has the dimension of approximately $4.2 \times 4.2m^2$ and a height of about $3.5m$. Therefore, the workspace is quite small when considering the power generated by the 6 propellers of the Firefly, with the effect of producing turbulences which is one source of disturbances during the real tests.

The World frame has the x-axis pointing toward the wall, the y-axis pointing right and the z-axis pointing down. Its origin is $2.5m$ away from the wall, in front of the marker in the third column and the third row from the top.

Given the limited dimensions of the real scenario and the uncertainties in pose estimation, experiments with the real platform have been focused mainly on obstacle avoidance tasks where a predetermined straight line is modified by the presence of one or more obstacles, while simulations have been performed in more complex scenarios, with nested obstacles and different paths. Tests in simulation also show that, given a path from a starting point to a goal, the appearance of obstacles can only deform the path but it cannot disconnect it: therefore, it is always possible to find a path between the start and the goal, also in complex environments.

5.2 Experiments in the Real Scenario

Experiments with the AscTec Firefly have been performed with different configurations of obstacles: some examples are shown in the Figs. 7, 8, and 9.

In each Figure, there are 3 different curves in different colors:

– green for the "ideal path", corresponding to the deformed curve generated by the planner (24), taking into account the surrounding obstacles;

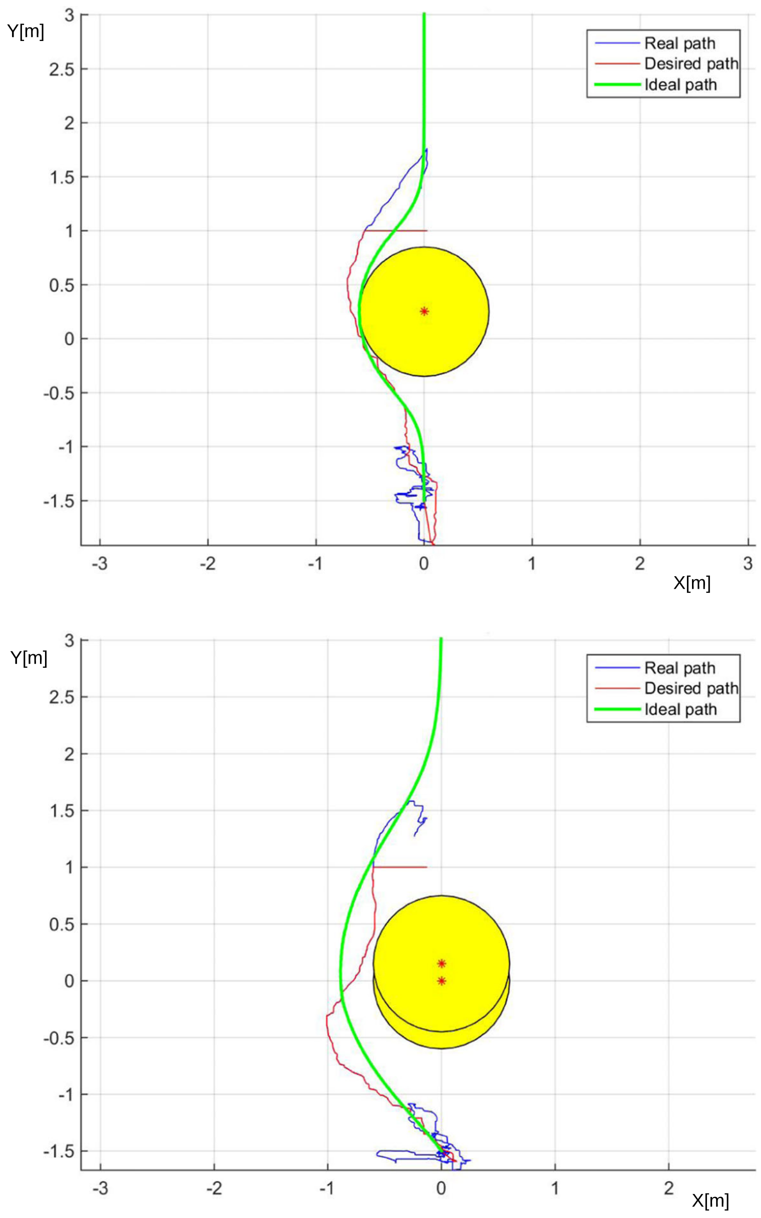
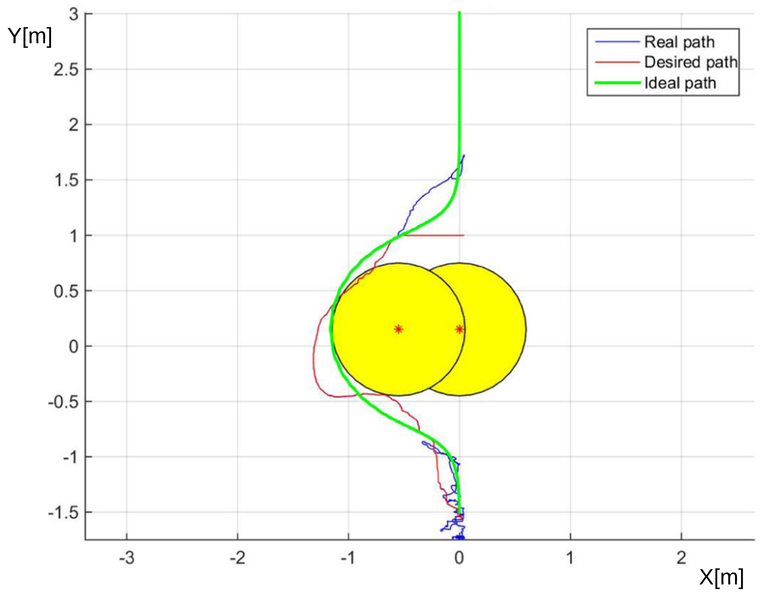


Fig. 7 Flying in straight line with 1 obstacle at (0.25,0), $R = 0.6m$. Up: $\sigma = R$; Down: $\sigma = 2R$



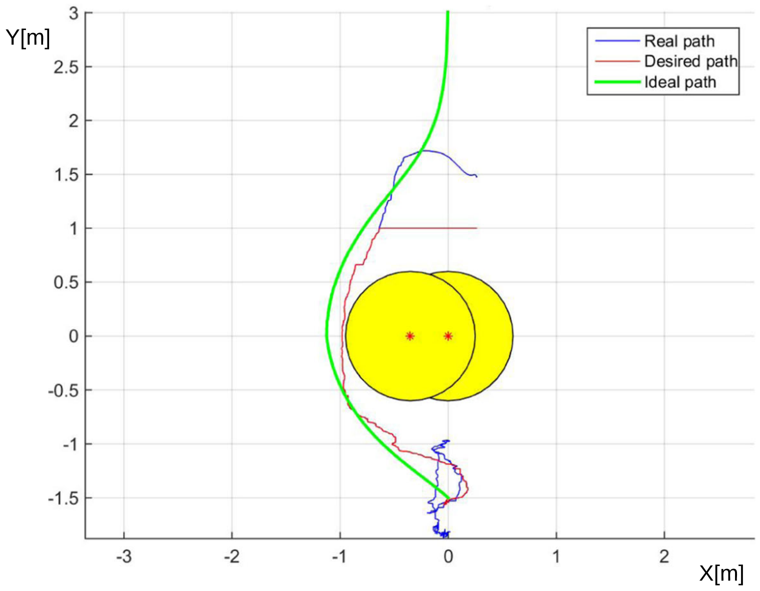
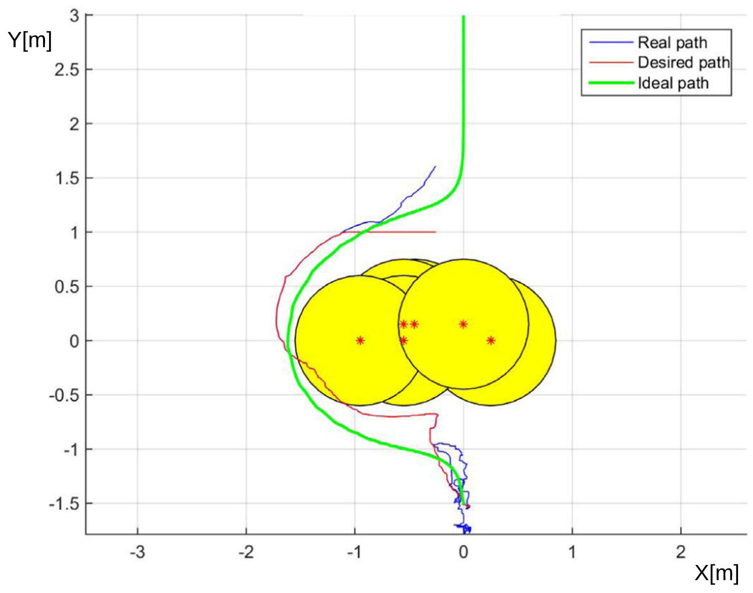


Fig. 8 Flying in straight line with 2 obstacles, $R = 0.6m$. Up: $\sigma = R$; Down: $\sigma = 2R$

– red for the "desired path", generated by the path generator to be fed into the controller (25);
– blue for the "real path" corresponding to the robot's actual motion.

All figures show obstacles as circles: the radius R of circles has been properly computed to take into account the dimensions of the hexacopter. That is, as long as the path does not intersect circles, it means that obstacles have been successfully avoided.

Experiments shows that, by tuning σ in Eq. 23, different run-time behaviours can be achieved (usually, a smoother and safer obstacle avoidance behaviour is achieved when σ is higher). Notice that, since obstacles are perceived in run-time, the same obstacle is perceived as a number of partially overlapping obstacles in some experiments due to perceptual errors in localizing the robot or the obstacles themselves (Fig. 7-Down, Fig. 9-Up). Notice also that, since the flying area is quite small, a person with safety



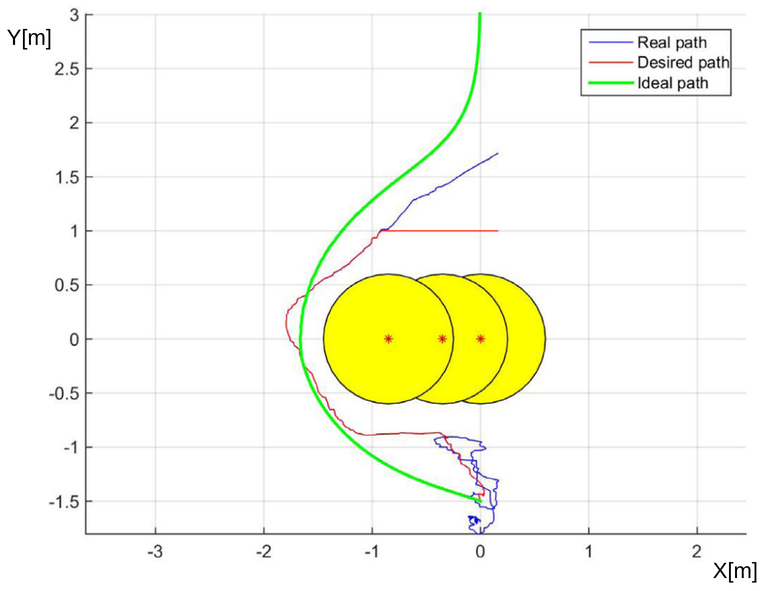


Fig. 9 Flying in straight line with 3 obstacles, $R = 0.6m$. Up: $\sigma = R$; Down: $\sigma = 2R$

equipment promptly grabs the robot after it avoids obstacles, and does not allow it to continue along the path. As a result, the desired and the real paths look truncated at the end of each experiment (the last part of the path, after avoiding obstacles, should be ignored in all the Figures).

Examining the results, the first aspect that should be underlined is that the "real paths" and the "desired paths" of the hexacopter coincide (except the last portions of all experiments, as noted); this straightforwardly follows the fact that the desired path is recalculated at each iteration in relation to the position of the quadrotor, according to Eq. 25.

In some experiments, e.g. Figure 7-Up or Fig. 8-Up, the robot was able to avoid obstacles but did not strictly follow the ideal path. This aspect is mainly due to the disturbances in the localization system (pose, attitude angles) leading to overshoots in the reaction of the robot keeping track of the ideal path. It should be remarked again that all perceptual information are acquired using on board sensors, and - similarly - all computations for path planning and control are performed onboard. However, after the overshoots, when the contribution of the gradient vector (mentioned in the Section 4.3) is big enough, the robot is driven back to the ideal path (Fig. 8-Up).

From the Figures it can also be noticed that a higher value of σ allows the multicopter to comfortably avoid all perceived obstacles, without any collision (see Figs. 7-Down, 8-Down and 9-Down, where $\sigma = 2R$). On the opposite, with a lower σ ($\sigma = R$ in Fig. 9-Up), the robot comes dangerously close to the obstacles.

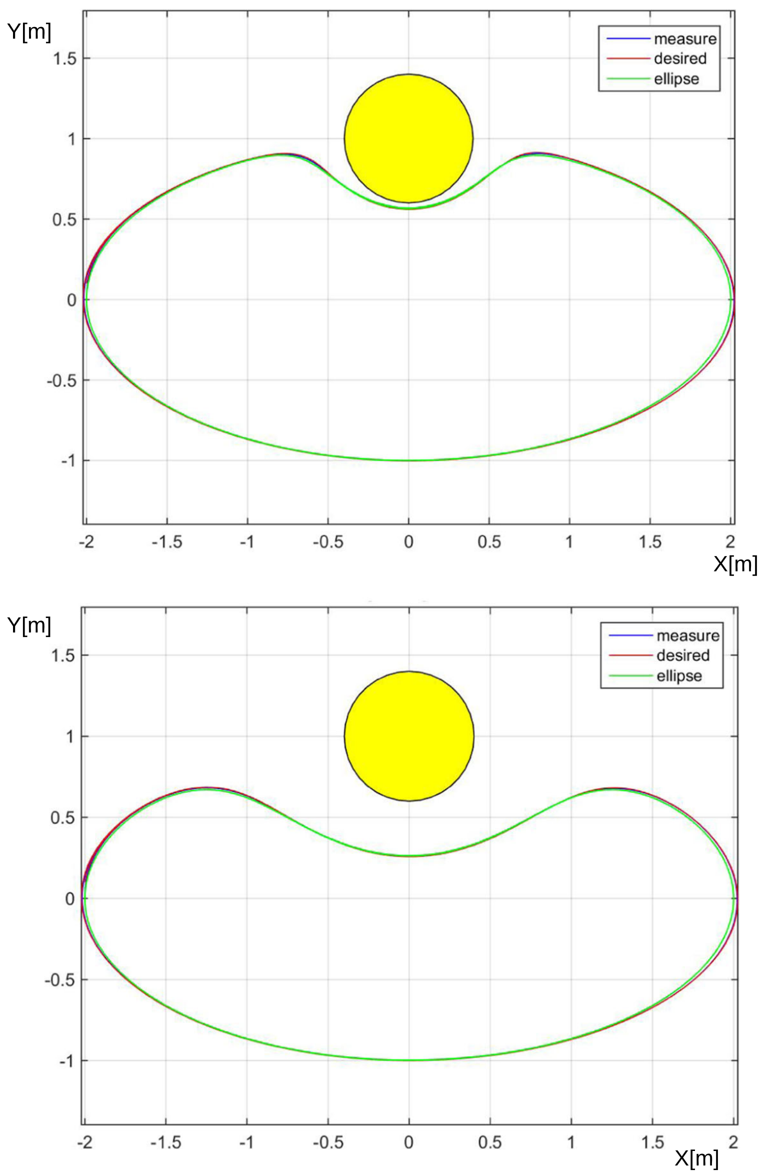


Fig. 10 Simulation of a multicopter following an ellipse with 1 obstacle (radius R = 0.4m) at (0,1), $k_{grad} = 2$, $k_V = 10$. Up: $\sigma = R$; Down: $\sigma = 2R$

Experimental tests confirm the fact that, by increasing the value of σ, the path modification will take place earlier, generating a safer path. For instance, in the conditions of a real implementation (in which disturbances are necessarily present) choosing a high value for σ, e.g. $\sigma = 2R$, can help the multicopter to avoid obstacles regardless any disturbances. The exact value of this parameter depends on the uncertainties of the localization system as well as the quality of the controller (both low and high level) and can be a topic for further studies.

Moreover, the estimation of the value of σ is also related to the speed of the robot. The described experiments have been performed at a low speed, i.e. the sum of the tangential and gradient component of the velocity was normalized to 0.1 m/s. This speed has been

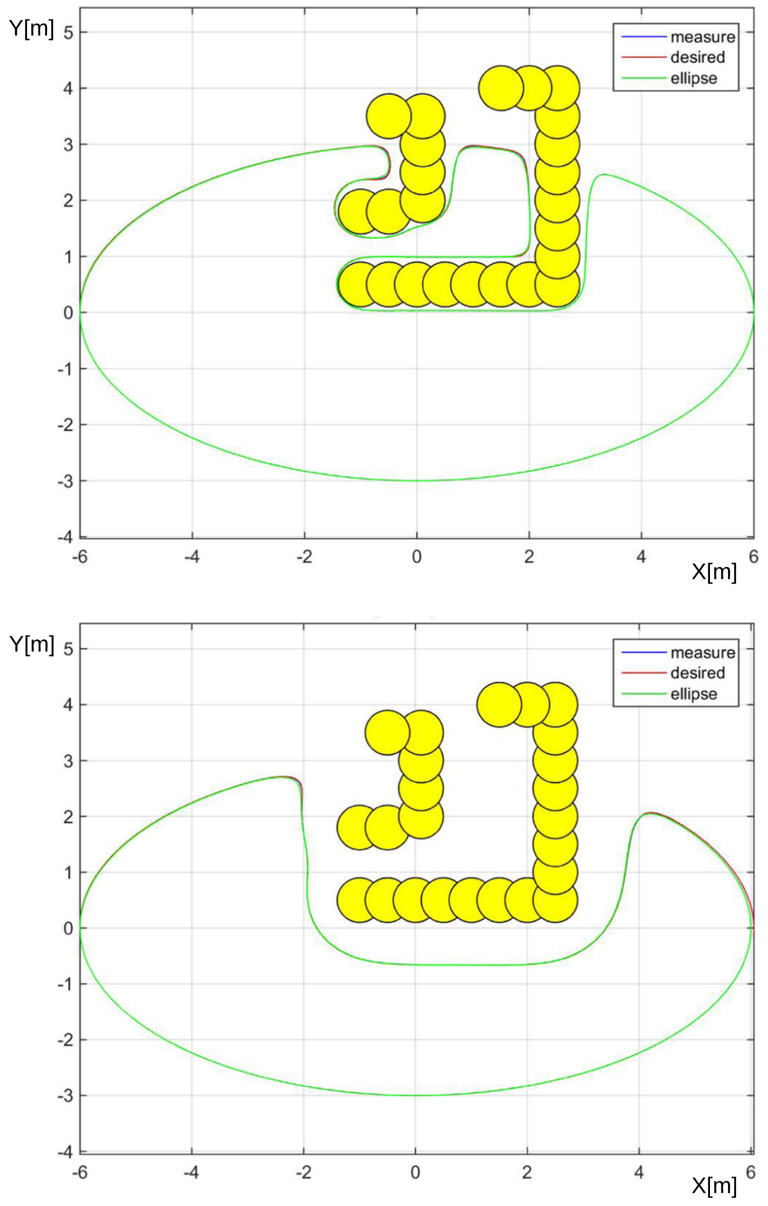


Fig. 11 Simulation of a multicopter following an ellipse with 2 nested horseshoes of obstacles (radius R = 0.4m) at (0,1), $k_{grad} = 2$, $k_V = 10$. Up: $\sigma = R$; Down: $\sigma = 2R$

chosen for safety reason, given the indoor environment and the errors of the localization system. However, the method can still be performed in real-time at higher speed: in this case, since higher deviations from the defined path are possible, σ should be increased for generating a safer trajectory. More details will be given in the next section.

5.3 Tests in Simulation

Simulated experiments have been performed in different, possibly maze-like scenarios in presence of multiple obstacles. The first two examples are shown in Figs. 10 and 11, that refer to simulated experiments performed with the Matlab/Simulink environment. As in the previous case, Figures are expressed with three different colors (green for the ideal path, red for the desired path and blue for the real path) and obstacles are represented as circles with radius R. Since the method is not probabilistic, a single run in simulation for each different scenario is sufficient for demonstrating its effectiveness.

The results show that the proposed controller and path generator can complete the tasks, i.e., the multicopter follows the "ideal path", while avoiding all encountered obstacles. Even in the cases of many obstacles continuously appearing or disappearing, the robot is able to modify its path, follow the contour of obstacles, and then return back to the original path.

Comparing Figs. 10-Up with 10-Down and Figs. 11-Up with 11-Down, it can be seen again how, by choosing a "suitable" value for σ in the obstacle Gaussian function (23), it is possible to change the shape of the deformed path in order to avoid obstacles. In particular, Fig. 11 shows that, by increasing the value of σ, the deformation of the path is such

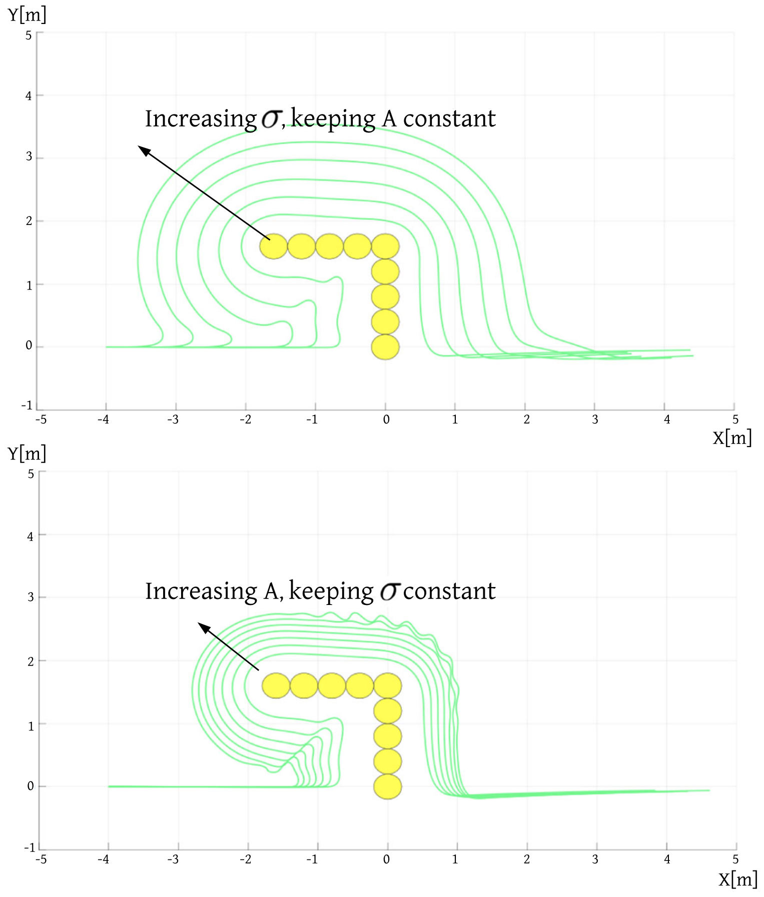


Fig. 12 Effect of the variation of the parameters σ (*up*) and A (*down*) of the obstacle functions, with the robot (*in the simulation environment*) moving on a straight line and encountering a L-shaped obstacle

that the robot completely avoids the horseshoe-shaped obstacles, while with a smaller σ the path is closer to the original ellipse, but is partially inside the nested obstacles. Again, the choice of σ depends on the context of application: a small value of σ generates a riskier path, but more similar to the predetermined one, while a higher value of σ leads to safer trajectories, but the deformation of the original curve is more relevant. It can be also noticed (Fig. 11) that the presence of many obstacles within a given path, even complex, between a starting point and a goal, does not disconnect the path but it just deform it: therefore, the proposed method ensures the possibility fo find a connected path between two points in space even in complex environments [41]. This aspect will be analyzed more in details in next section, also in relation to the other aforementioned methods for obstacle avoidance.

Further simulated tests have been performed using the ROS-Gazebo environment. This simulation environment has been chosen because it offers a detailed model of the hexacopter (Asctec firefly) used for the experimental phase, and for the possibility of clearly visualizing the 3D-path of the simulated UAV. Figure 12 shows the effect of the variation of σ in comparison with the variation of A, the amplitude of the Gaussian obstacle functions. In this experiment, the robot is following a straight line at a constant height and a L-shaped obstacle is placed in the middle of the trajectory. The results confirm that the robot can successfully overcome the obstacles even with complex environmental configurations, and that increasing σ allows the robot to fly more distant from the obstacles, generating a trajectory that is safer but more distant from the original one. In particular, it can be seen that while the variation of σ causes a well-tunable and controllable behaviour of the robot (Fig. 12-up), varying A has a lower impact on the resulting path.

Indeed, increasing the amplitude of the obstacle functions has a more limited effect on the variation of the trajectories, and moreover it can be observed that, when the value of A becomes too high, some oscillations can arise in the motion of the UAV, with possible dangerous effects. This can be due to the fact that, increasing A and keeping σ constant, the gradient component of the velocity vector related to the obstacle function becomes too high when the robot is inside the radius of influence of the obstacle, with a resulting

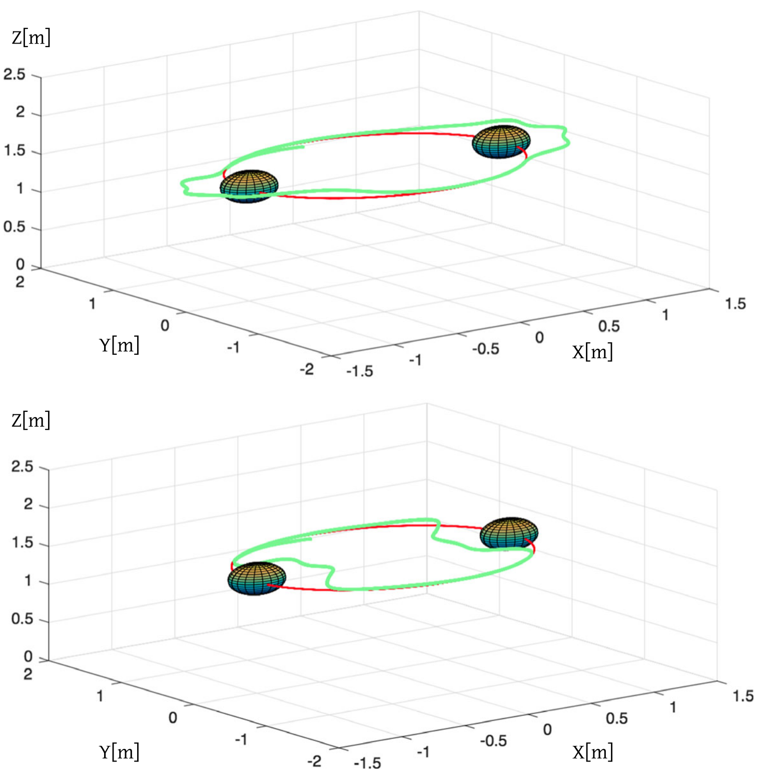


Fig. 13 Simulated path of the UAV, flying on a circular path with two circular obstacles placed on the path. The obstacles can be avoided flying externally (*up*) or internally (*down*) with respect of the obstacles. The green line represents the actual path of the robot, the red line the path without obstacles

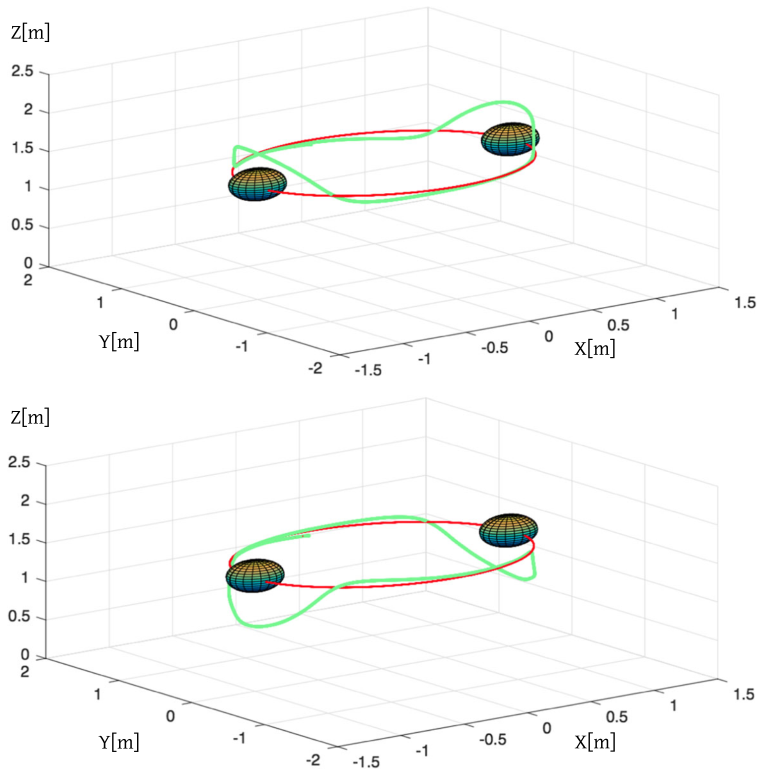


Fig. 14 Simulated path of the UAV, flying on a circular path with two circular obstacles placed on the path. The obstacles can be avoided flying above (*up*) or below (*down*) the obstacles. The green line represents the actual path of the robot, the red line the path without obstacles

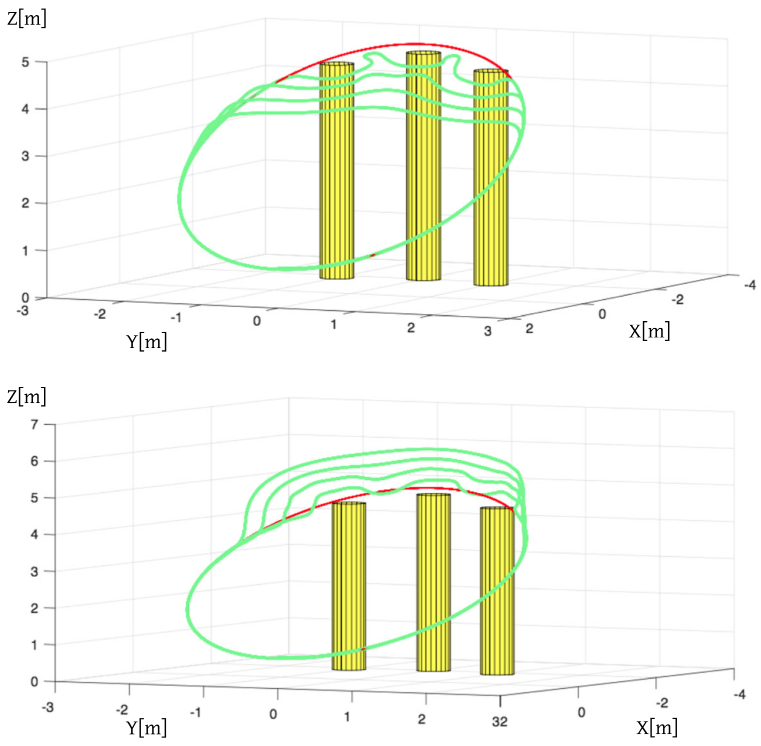


Fig. 15 Simulated path of the UAV, flying on an elliptical path with three cylindrical obstacles placed on the path. The obstacles are avoided adding the obstacle functions to f_1, and thus with the UAV flying internally with respect of the obstacles (*up*) or adding the obstacle functions to f_2, and thus with the UAV going above the obstacles (*down*). The green lines represents the actual path of the robot (four different runs varying σ), the red line the path without obstacles

unstable and unpredictable behaviour. For this reason, while using Gaussian functions as obstacle function, tuning the parameter σ is not only sufficient but also preferable for the implementation of the method.

Moreover, the simulation environment has been used in order to investigate the possibility of exploiting the 3D motion capabilities of the UAV implementing the described approach. First of all, how it has been mentioned in Section 4.2, by adding the obstacle functions to f_1 or f_2 it is possible to overcome the obstacle in the xy plane or along the z direction (assuming that the gradient ∇f_1 is never parallel to the z axis of the world frame and that the gradient ∇f_2 has a non-null component along the z axis). This aspect has been practically tested by making the robot following a circular trajectory at constant height, defined as the intersection of a cylinder and a plane parallel to the xy plane. Two spherical obstacles have been placed on the trajectory, and the two related obstacle functions have been added in four different modalities: *a)* added to f_1 with positive sign, *b)* added to f_1 with negative sign, *c)* added to f_2 with positive sign, *d)* added to f_2 with negative sign. The effects are shown in Fig. 13 (cases *a)* and *b)*, in the xy plane) and Fig. 14 (cases *c)* and *d)*, along the z direction). It can be seen how the proposed approach allows the user to modify the resulting path (avoiding the obstacles) in a very simple fashion. The shown results are obtained with the UAV moving at a speed of 0.2 m/s.

Further tests have been aimed at testing the same aspect with a 3D trajectory, in which the robot was moved along an elliptical path, defined as the intersection of a cylinder and the plane $x + z = k$. In this case, the resulting path does not line on a plane parallel to the xy plane, therefore the UAV will move on the path varying its altitude. However, the same considerations as before are still valid: the modulation of σ helps in obtaining different trajectories, and the obstacles can be avoided along different directions with respect of the function chosen for the obstacle avoidance task. Both aspects are shown in Fig. 15 (also in this case the speed of the robot is 0.2 m/s).

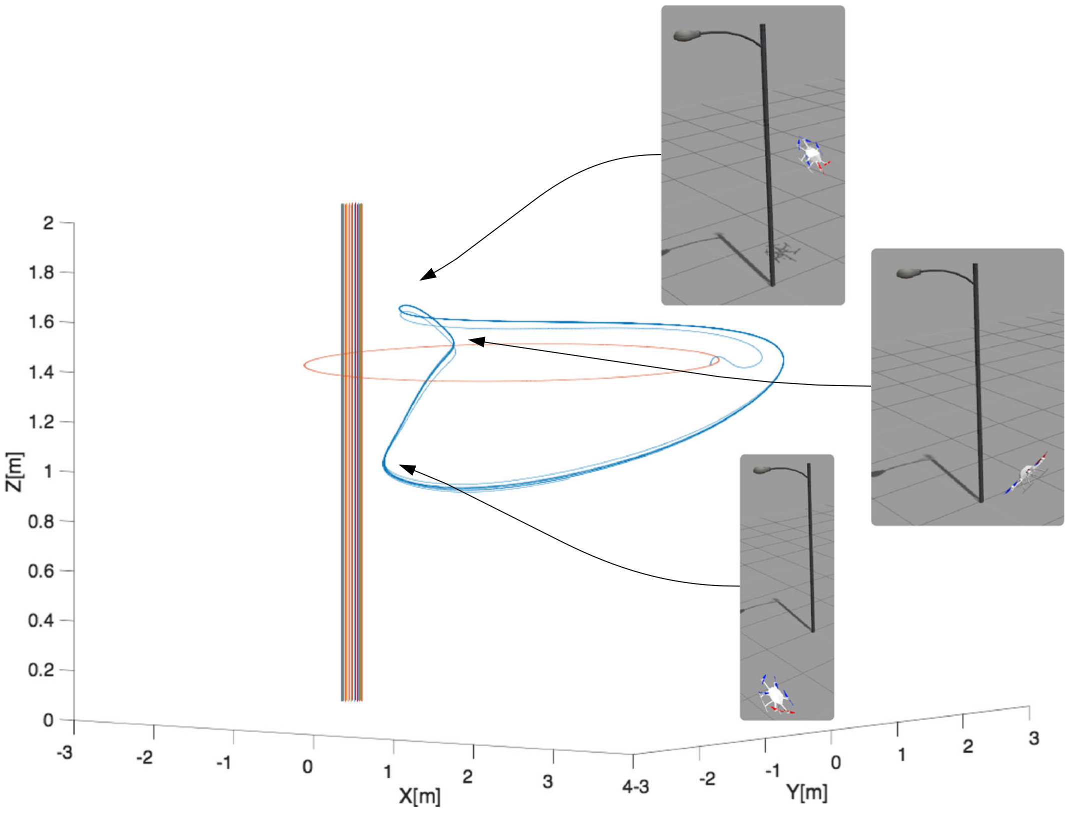


Fig. 16 Simulated path of the UAV flying at high speed (4 m/s) on a circular trajectory at constant height, with a cylindrical obstacle placed on the path

Finally, simulation experiments have been aimed at analyzing the implementation of the approach with the robot running at high speed. In fact, while tests with the real robot have been conducted at a safe speed of 0.1 - 0.2 m/s (as well as for tests in simulation), it is extremely important that this method could be implementable even in more realistic scenarios, i.e. when the robot is moving faster. At this aim the hexacopter has been flied at a speed of 4 m/s, while moving on a circle of radius of 2 meters, and facing an obstacle placed on the path. It can be seen how the approach allows the robot to safely avoid the obstacle, while following the circular trajectory (Fig. 16). Obviously the parameters of the obstacle functions should be opportunely tuned: in particular the value of σ has been considerably increased with respect of the previous runs. It is worth saying that the detection of the obstacles has not been evaluated in the simulation experiments, since it was not the object of this work; of course, when flying at high speed, this can be a more critical aspect.

5.4 Comparison with other Approaches

In all shown results, in particular in the complex maze-like scenarios (e.g. Figure 11), it can be seen how the robot never gets stuck in the maze created by obstacles. Indeed the algorithm always manages to drive the robot back to the original path, after obstacle avoidance. This represents a great advantage in comparison to other methods, such as the Artificial Potential Field approach [26, 27]: with the proposed method a maze-like situation does not produce any local minima.

The described strategy presents advantageous properties also when compared with the other techniques usually adopted for path planning of UAVs. With references to methods for obstacle avoidance seen in section 2, it should be here underlined how the proposed methodology avoids the calculation of motion constraints and actuator limitations that are not easy to set and should be calculated for [29], while it just requires the resolution of a simple system of two equations instead of optimizing a 9^{th} order polynomial [28] or finding a trajectory between keyframes that minimizes the integral of the norm of the jerk (the derivative of the acceleration) [31].

It comes then straightforwardly that while the methods described in [28, 29] and [31] can difficultly be applied to dynamic environments, the proposed approach can perfectly work with moving obstacles, since its limited computational complexity. Moreover, the current method does not require to set a fixed time to travel along the trajectory, as in [28]. Finally the method gives an integrated approach taking into account the dynamics of the robot and the structure of the environment, differently from [30] that completely ignores vehicle dynamics; and it allows users to fly the robot with different altitudes and different headings, differently from [32].

6 Conclusions

Given the wide usage of multirotors in contexts such as disaster scenarios, the possibility to let these UAVs move in an autonomous fashion, avoiding detected obstacles, is of the utmost importance. Therefore, this work was focused on the analysis of a novel technique for path following and obstacle avoidance, applied to multirotors.

The approach is based on the mathematical equations of the multicopter's dynamic model and proposes a model-based controller and path generator that can be applied to different multirotors (i.e., it is not limited to hexacopters, used in real experiments).

In relation to the control of the robot, a Cascaded control architecture based on the Feedback linearization technique was analyzed, that allowed the robot to change its position continuously without changing the modes of the controller. The control strategy was implemented together with a novel method to generate the motion paths in run-time taking into account the obstacles perceived in the environment. Both the controller and the path generator provide some degrees of freedom to tune the run-time performance of the multirotor.

Experiments, both in simulation and with a real robot, have been performed to validate the approach, giving encouraging results. In particular, results show that the method allows to effectively avoid obstacles and that tuning some parameters different run-time behaviors can be achieved. Even in cases of many obstacles and complex scenarios (tested only in simulation), the proposed approach ensures that the robot will never get stuck into local minima.

Further improvements, such as the execution of tests in real scenarios with more complex paths and a higher number of obstacles, and the adoption of visual SLAM techniques and high level planners (to tune the parameters and to reason about the path), are planned for the next future.

References

1. Pratt, K.S., Murphy, R., Stover, S., Griffin, C.: CONOPS and autonomy recommendations for VTOL small unmanned aerial system based on Hurricane Katrina operations. J. Field Rob. **26**(8), 636–650 (2009)
2. Murphy, R.R., Steimle, E., Griffin, C., Cullins, C., Hall, M., Pratt, K.: Cooperative use of unmanned sea surface and micro aerial vehicles at Hurricane Wilma. J. Field Rob. **25**(3), 164–180 (2008)
3. Baiocchi, V., Dominici, D., Milone, M.V., Mormile, M.: Development of a software to plan UAVs stereoscopic flight: An application on post earthquake scenario in L'Aquila city. In: Computational Science and Its Applications-ICCSA 2013, pp. 150–165. Springer, Berlin Heidelberg (2013)
4. Qi, J., Song, D., Shang, H., Wang, N., Hua, C., Wu, C., Han, J.: Search and Rescue Rotary-Wing UAV and Its Application to the Lushan Ms 7.0 Earthquake. J. Field Rob. (2015)
5. Recchiuto, C.T., Sgorbissa, A., Zaccaria, R.: Visual feedback with multiple cameras in a UAVs Human-Swarm Interface. Robot. Auton. Syst. **80**, 43–54 (2016)
6. Apvrille, L., Roudier, Y., Tanzi, T.J.: Autonomous drones for disasters management: Safety and security verifications. In: 2015 1st 1000 URSI Atlantic Radio Science Conference (URSI AT-RASC), pp. 1–2. IEEE (2015)
7. Ramírez, A., Espinoza, E.S., Carrillo, L.R.G., Mondie, S., Lozano, R.: Stability analysis of a vision-based UAV controller for autonomous road following missions. In: 2013 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1135–1143. IEEE (2013)
8. Bouabdallah, S.: Design and control of quadrotors with application to autonomous flying. (Doctoral dissertation, Ecole Polytechnique Federale de Lausanne) (2007)
9. Altuğ, E., Ostrowski, J.P., Mahony, R.: Control of a quadrotor helicopter using visual feedback. In: Proceedings of the International Conference on Robotics and Automation, 2002, ICRA'02. IEEE Vol. 1, pp. 72–77. IEEE (2002)
10. Mistler, V., Benallegue, A., M'sirdi, N.K.: Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In: Proceedings of the 10th IEEE International Workshop on Robot and Human Interactive Communication, 2001, pp. 586–593. IEEE (2001)
11. Xu, R., Özgüner, Ü.: Sliding mode control of a quadrotor helicopter. In: 2006 45th IEEE Conference on Decision and Control, pp. 4957-4962. IEEE (2006)
12. Bouffard, P., Aswani, A., Tomlin, C.: Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 279-284). IEEE (2012)
13. Alexis, K., Papachristos, C., Nikolakopoulos, G., Tzes, A.: Model predictive quadrotor indoor position control. In: 2011 19th Mediterranean Conference on Control & Automation (MED), pp. 1247–1252. IEEE (2011)
14. Bangura, M., Mahony, R.: Real-time model predictive control for quadrotors (2014)
15. Castillo, P., Dzul, A., Lozano, R.: Real-time stabilization and tracking of a four-rotor mini rotorcraft. IEEE Trans. Control Syst. Technol. **12**(4), 510–516 (2004)
16. Achtelik, M., Bierling, T., Wang, J., Hocht, L., Holzapfel, F.: Adaptive Control of a Quadcopter in the Presence of large/complete Parameter Uncertainties. Infotech Aerospace, 2011–1485 (2011)
17. Achtelik, M.W., Lynen, S., Chli, M., Siegwart, R.: Inversion based direct position control and trajectory following for micro aerial vehicles. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2933-2939. IEEE (2013)
18. Kim, H.J., Shim, D.H.: A flight control system for aerial robots: algorithms and experiments. Control. Eng. Pract. **11**(12), 1389–1400 (2003)
19. Tabatabaei, S.A.H., Yousefi-Koma, A., Ayati, M., Mohtasebi, S.S.: Three dimensional fuzzy carrot-chasing path following algorithm for fixed-wing vehicles. In: 2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM), pp. 784-788. IEEE (2015)
20. 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)
21. Frazzoli, E., Dahleh, M.A., Feron, E.: Trajectory tracking control design for autonomous helicopters using a backstepping algorithm. In: Proceedings of the American Control Conference, 2000, Vol. 6, pp. 4102–4107. IEEE (2000)
22. Nelson, D.R., Barber, D.B., McLain, T.W., Beard, R.W.: Vector field path following for miniature air vehicles. IEEE Trans. Robot. **23**(3), 519–529 (2007)
23. Kothari, M., Postlethwaite, I., Gu, D.W.: UAV path following in windy urban environments. J. Intell. Robot. Syst. **74**(3–4), 1013–1028 (2014)
24. Osborne, J., Rysdyk, R.: Waypoint guidance for small UAVs in wind. AIAA Infotech Aerospace **193**(1–4), 1–12 (2005)
25. Alexis, K., Nikolakopoulos, G., Tzes, A.: On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances. Asian J. Control **16**(1), 209–224 (2014)
26. Rezaee, H., Abdollahi, F.: Adaptive artificial potential field approach for obstacle avoidance of unmanned aircrafts. In: 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1-6 IEEE (2012)
27. Nieuwenhuisen, M., Droeschel, D., Schneider, J., Holz, D., Labe, T., Behnke, S.: Multimodal obstacle detection and collision avoidance for micro aerial vehicles. In: 2013 European Conference on Mobile Robots (ECMR), pp. 7–12. IEEE (2013)
28. Achtelik, M.W., Weiss, S., Chli, M., Siegwart, R.: Path planning for motion dependent state estimation on micro aerial vehicles. In: 2013 IEEE International Conference on

Robotics and Automation (ICRA), pp. 3926-3932). IEEE (2013)

29. Webb, D.J., van den Berg, J.: Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 5054-5061. IEEE (2013)
30. He, R., Prentice, S., Roy, N.: Planning in information space for a quadrotor helicopter in a GPS-denied environment. In: IEEE International Conference on Robotics and Automation, 2008. ICRA 2008, pp. 1814-1820). IEEE (2008)
31. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 2520-2525. IEEE (2011)
32. Vista IV, F.P., Singh, A.M., Lee, D.J., Chong, K.T.: Design convergent Dynamic Window Approach for quadrotor navigation. Int. J. Precis. Eng. Manuf. **15**(10), 2177–2184 (2014)
33. Mullhaupt, P., Srinivassn, B., Bonvin, D.: A Two-time-scale Controller for a Differentially Cross-coupled System. In: Proceedings of the American Control Conference, 1997, Vol. 6, pp. 3839-3841. IEEE (1997)
34. Wagner, K.G.: Nonlinear noninteraction with stability by dynamic state feedback. SIAM J. Control. Optim. **29**(3), 609–622 (1991)
35. Murray, R.M., Li, Z., Sastry, S.S., Sastry, S.S.: A mathematical introduction to robotic manipulation. CRC press (1994)
36. Khalil, H.K., Grizzle, J.W.: Nonlinear systems, vol. 3. Prentice hall, New Jersey (1996)
37. Schmitendorf, W.E., Stalford H. L.: Improving Stability Margins via Dynamic-State Feedback for Systems with Constant Uncertainty. IEEE Trans. Automat. Contr. **42**, 1161–1163 (1997)
38. Zhang, C., Fu, M.: A Revisit to the Gain and Phase Margins of Linear Quadratic Regulators. IEEE Trans. Automat. Contr. **41**, 1527–1530 (1996)
39. Morro, A., Sgorbissa, A., Zaccaria, R.: Path following for unicycle robots with an arbitrary path curvature. IEEE Trans. Robot. **27**(5), 1016–1023 (2011)
40. Sgorbissa, A., Zaccaria, R.: 3D path following with no bounds on the path curvature through surface intersection. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4029-4035. IEEE (2010)
41. Planning, I.ntegrated.R.obot., Avoidance, O.bstacle.: Path Following and Control in 2D and 3D: a case study with UGV, UUV, and UAV. Submitted to The International Journal of Robotics Research (2016)
42. Siegwart, R., Nourbakhsh, I.R., Scaramuzza, D.: Introduction to autonomous mobile robots. MIT press (2011)
43. Simulink, M., Natick, M.A.: The Mathworks (1993)
44. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 (IROS 2004), Vol. 3, pp. 2149-2154. IEEE (2004)
45. Furrer, F., Burri, M., Achtelik, M., Siegwart, R.: RotorS. A Modular Gazebo MAV Simulator Framework. In: Robot Operating System (ROS), pp. 595-625. Springer International Publishing (2016)
46. Piaggio, M., Sgorbissa, A., Zaccaria, R.: A programming environment for real-time control of distributed multiple robotic systems. Adv. Robot. **14**(1), 75–86 (2000)
47. Recchiuto, C., Sgorbissa, A., Wanderlingh, F., Zaccaria, R.: UAV Teams In Emergency Scenarios: A Summary Of The Work Within The Project PRISMA. In: Proceedings of the 2nd Italian Workshop on Artificial Intelligence and Robotics. CEUR Workshop. Vol. 1544. Ferrara, Italy (2015)
48. Munoz-Salinas, R.: ARUCO: a minimal library for Augmented Reality applications based on OpenCv. Universidad de Cordoba (2012)

Phuong D. H. Nguyen has attended Istituto Italiano di Tecnologia (IIT) since January, 2016 as PhD Fellow and focuses on research of perceptual and sensorimotor capabilities for humanoid robotics. He also holds Marie Curie Fellowship as an Early Stage Researcher. He got Master degree in Advanced Robotics from Ecole Centrale de Nante, France and Università degli Studi di Genova, Italy, where he conducted research on Control and Planning for UAV robots.

Carmine T. Recchiuto received his Master Degree in Electronic Engineering from the University of Pisa in 2008. From 2009 to 2012 he was Research Fellow at the Scuola Superiore Sant'Anna, where he was involved in projects related to the realization of tactile sensors and humanoid robotics. From 2013 he was Research Fellow in the Department of Informatics, Bioengineering, Robotics, and System Sciences (DIBRIS), at the University of Genova, where he was involved in the PRISMA project, focussing his work in the implementation of UAVs in emergency scenarios. He received his Ph.D in 2015 in Microsystems from the University of Rome Tor Vergata. He is currently Post-Doc Fellow with the University of Genova. His main interests are in the fields of tactile sensors, UAVs, humanoids and human-robot interaction.

Antonio Sgorbissa is Associate Professor at University of Genova, Department of Informatics, Bioengineering, Robotics, and Systems Sciences (DIBRIS), Italy. In 2000 he received his Ph.D. in Robotics from the University of Genova, and from 2001 to 2004 he was a Post Doc at Georgia Tech, University of Parma and later at University of Genova. He currently teaches Ambient Intelligence, Cooperative Robotics, and Real-Time Operating Systems at the Faculty of Engineering as well as in the European Master on Advanced Robotics (EMARO), and Computer Science at the Faculty of Psychology. His main research interests are in the area of multi-robot systems, wearable robotics, and ambient intelligence. His research interests include also planning, knowledge representation, and machine consciousness. He is author/coauthor of more than 100 international scientific papers, and of two international patents in Robotics.