Backstepping control law application to path tracking with an indoor quadrotor

Maxime Lecointe, Francois Defay, and Caroline P. Carvalho Chanel

Abstract

This paper presents an application of the backstepping control to a path tracking mission using an indoor quadrotor. This study case starts on modeling the quadrotor dynamics in order to design a backstepping control which we applied directly to the Lagrangian dynamic equations. The backstepping control is chosen due to its applicability to this class of nonlinear and under-actuate system. To test the designed control law, a complete quadrotor model identification was performed, using a motion capture system. The procedure used to obtain a good model approximation is presented. Experimental results illustrate the validity of the designed control law, including rich simulations and real indoor flight tests.

1 Introduction

Remotely Piloted Aircraft Systems (RPAS) have been used in many civil or military applications. One can cite missions like : surveillance, search and rescue, target tracking, cartography, etc. To well fulfill the mission, it is mandatory the RPAS would be able to move in a constraint flight space following some complex path. The quality of the movement depends on a lot of parameters and functions. One can cite : the control law used to stabilize and to move the RPAS to some position; the navigation component used to track the state of the system; the guidance component used to infer input references to the control depending on the environment constraints and on the motion planning; the mission planning which defines the or-

M. Lecointe · F. Defay · C. P. Carvalho Chanel
Université de Toulouse
ISAE-SUPAERO Institut Supérieur de l'Aéronautque et le l'Espace
DMIA - Department of mathematics, computer science and automatic control
10 av. Edouard Belin - BP 54032 - 31055 TOULOUSE Cedex 4 - FRANCE
e-mail: max.lec.ml@gmail.com, francois.defay@isae-supaero.fr, caroline.chanel@isae-supaero.fr

der of movements to reach the mission goal, and which quality and efficiency is answerable to how good the GNC (guidance, navigation and control) components work.

The attitude control and modeling of a quadrotor is studied since many years [1],[7], one can notice that full control is not currently used on quadrotor application. Some papers, as [3] deals with, but the applicative case on real testbed is not a lot published. Today, the classical approach (cascade loop for attitude, navigation), as introduced by *Hamel et al.* [7], supposes that roll, pitch, yaw axes are not coupled. Backstepping approach is used since a long time for quadrotor in simulation [2],[4],[9] but few papers deals on the applicative case, which is the aim of this paper.

In this paper, the main interest is to provide an easy way to track a path for a quadrotor for indoor flight purpose. For this application, the classical PID approach for attitude stabilization is chosen because of its good performance, stability and robustness. Trajectory or path tracking is more and more studied since quadrotor's open source projects and friendly testbed are available. Some control laws have already been used [11], but few of them present experimental results [13]. Backstepping approach was chosen in this paper using the Lagrangian dynamics equations [5]. This paper presents a complete applicative case study of path tracking on a real testbed, using a complete quadrotor model in Simulink to tune the control parameters. The final interest on this UAV framework is to provide a stabilized UAV in order to achieve complex missions as target detection and formation flying in constraint environments.

The paper is organized as follows: Section 2 presents the experimental framework, the AR.drone's model and the attitude stabilization. In Section 3, the backstepping applicative case for path tracking is presented. Finally, Section 4 shows some experimental results to demonstrate the interest of the approach.

2 Quadrotor dynamics

2.1 Testbed framework

At ISAE-SUPAERO Institute, a complete framework has been developed to study UAV and robots applications. For students's works and research on guidance and planning, the AR.drone quadrotor developed by the Parrot Society is chosen. Therefore, only the mechanical basis, motors and propellers was conserved. All the embedded electronics used to control the UAV was changed by a Pixhawk device [10] for the attitude stabilization. There is an other electronic device, a Gumstix board, carried with a wifi key which is used to deploy the embedded robotic architecture composed of all necessary components, as for instance, the state estimation component. The high level Gumstix device is connected with the Pixhawk device by a serial connection, and with the ground station by a wifi connection. In the ground

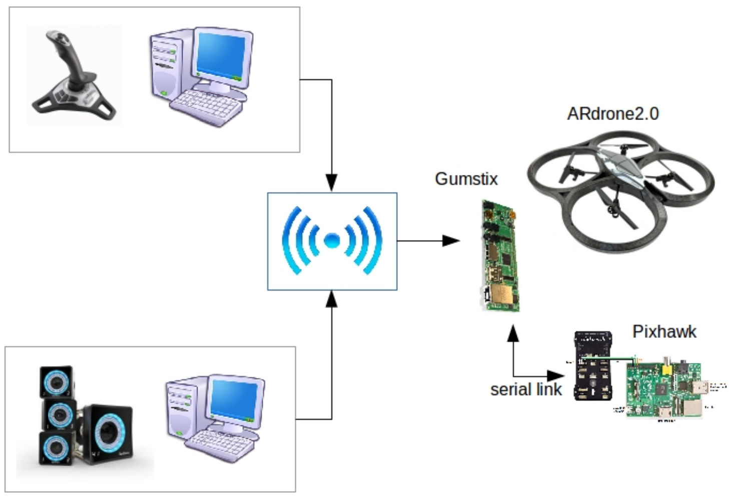


Fig. 1 Framework environment description.

station, the operator can chose between control himself with a joystick device the quadrotor or control the quadrotor using the automatic path generation and the backstepping control law.

The figure 1 shows a description of the test environment. The flight area, about $50m^2$, is equipped with an 'OptiTrack' motion capture system. It allows us to decouple the indoor localization problem and the control law design problem. The motion capture system is able to send directly to the Gumstix board the local position and speed with a frequency highest than $100Hz$. This information is treated by a special designed component on the robotic real time architecture.

The robotic real time architecture used is the Orocos Toolchain [6]. Orocos is based on modular and run-time configurable software components. In Orocos each component has an standard interface based on input and output ports for the data flow, properties and services. In our application case, communication with the motion capture system, state estimation and control are encapsulate in Orocos compo-

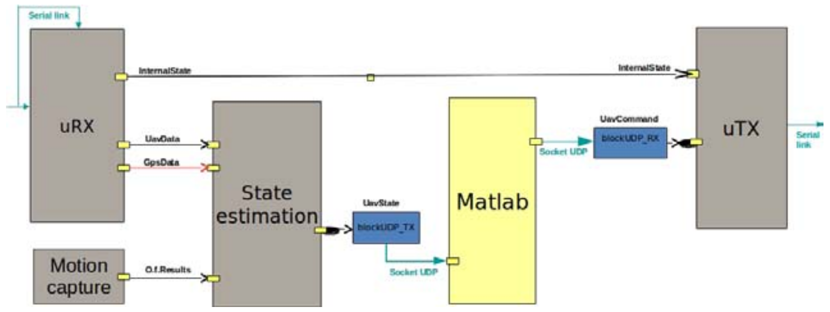


Fig. 2 Orocos components (in gray color), Matlab (in yellow), transmission blocks (in blue) schema.

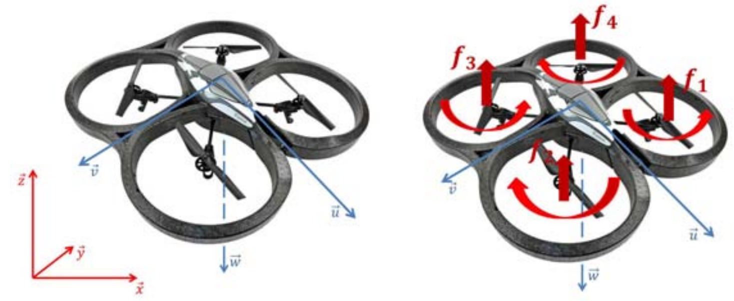
nents and connected between them. Orocos guarantee real time and safety communications between components.

The ground station is a computer carried with the joystick device and where the Matlab Real Time Windows Target application runs. Hence the operator can chose to control manually or automatically the quadrotor. The designed control law and path generation are computed in real time on Matlab with a frequency of $100Hz$. So all, the control laws are discretized by automatic code generation tools of Matlab/Simulink. Matalb/Simulink performs the human-machine interface and allows to directly obtain the results of the new control laws.

In the Figure 2 the Orocos components schema is shown. Note that Matlab real time target is used to read (UDP socket) the information coming from the state estimation component turning on the Gumstix board, and to send the computed command (UDP socket) to the control component running on the Gumstix board. For real flight tests, it was necessary to sample the control with the same period used by the communication system (send/reception of UDP pack). It is interesting to observe that the frequency used for communication is enough to guarantee no latency or excessive delays.

2.2 Quadrotor model

A quadrotor is a rotary wing aircraft composed by four rotors to its lift. These four rotors are generally placed on the cross extremities of the rigid structure of the aircraft. In the center of the rigid structure the embedded electronic for the control and battery are placed. To avoid the aircraft turns over itself, two rotors usually turn in clockwise sense and the others two rotors in counter-clockwise sense. Hence, to control the movement of the aircraft, it is usual to place each couple of rotors turning in the same sense in opposite way on the rigid cross structure. The Figure 3(b) illustrate this principle.



(a) Local and inertial frames for the ARdrone 2.0 structure used for this study.

(b) Lift forces schema.

Fig. 3 Illustrative schema of local and inertial frames, and lift forces for the quadrotor.

The generalize configuration of the quadrotor is $q = (x, y, z, \varphi, \theta, \psi)$, where (x, y, z) represents the relative position of the center of mass of the quadrotor, with respect to a local frame, and (ϕ, θ, ψ) are the three Euler angles representing the orientation of the rotorcraft, with respect to a local frame, namely yaw, pitch and roll. The translational and rotational variables are respectively defined as $\xi = (x, y, z)^T \in R^3$ and $\eta = (\varphi, \theta, \psi)^T \in R^3$.

The total mass of the quadrotor (m) corresponds to the sum of this rigid cross structure (m_c), the mass of the four rotors (m_i), considered as identical, and the mass of the embedded electronic and battery (m_b),

$$m = m_1 + m_2 + m_3 + m_4 + m_c + m_b \tag{1}$$

Considering that the quadrotor is perfectly symmetric, ignoring the Coriolis (centripetal) forces, and finally doing hypothesis that the effective inertia is constant, the inertia matrix J can be written as :

$$J = \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \tag{2}$$

where, I_x represents the inertia on the $\overrightarrow{v}$ axis which goes throw in the cross' center between the front and rear rotors, I_y represents the inertia over the $\overrightarrow{u}$ axis which goes throw the cross' center between the right and left rotors, and finally I_z represents the inertia over the $\overrightarrow{w}$ axis, perpendicular to the v and u plan (see Figure 2.2).

Note that only these physical parameters m and J are used to pose and solve mechanical equations of our control system. In this paper, this control scheme is dedicated for indoor navigation and planning study, one can assume that the relative ground speed is low, so this simplification is justified. We empathize that the simple mechanical model used in this study does not implicate our approach as it will be seen in the next sections.

The simple mechanical model equation can be written as:

$$m\ddot{\xi} + \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} = F_d \tag{3}$$

and, considering that the coriolis term is neglicted, the rotational subsystem is given by:

$$J\ddot{\eta} = \tau \tag{4}$$

where F_d and τ represents respectively the thrust force and the total torque.

In order to better design the control law to path tracking purpose, the aerodynamic forces and torques involved in our application have to be identified in order to find the link between these forces and the speed rotation of rotors.

Aerodynamic forces

The principle of the movement is simple, for example: pitch movement is obtained by increasing (resp. reducing) the speed of the rear motors while reducing, (resp. increasing) the speed motor of the front motors. The same idea can be used to understand the roll movement. The yaw one is obtained by increasing (resp. reducing) the rear and front speed motors as the same time as decreasing (resp. increasing) right and left speed motors.

More precisely, the aerodynamic forces can be divided into two projected forces related to two axes. First, the drag force is parallel to the rotation plan of rotors, i.e to the $\overrightarrow{v}$ and $\overrightarrow{u}$ axis. It corresponds to the drag drill when in quadrotor is in air, due to the mass air acceleration against to the blades of the rotors.

$$f_{d_i} = k_d \omega_i^2 \tag{5}$$

where f_{d_i} is the drag force of each rotor i ,k_d the drag coefficient, and w_i the rotation speed of each rotor i.

Secondly, the lift force can be observed which is conduct in the perpendicular axis with respect to the rotor rotation plan. This force is responsible for the weight compensation and its enable the quadrotor to remain in air.

$$f_i = k_\omega \omega_i^2 \tag{6}$$

where , f_i represents the lift force of each rotor i, k_w represents the thrust coefficient, and w_i the speed rotation of the rotor i.

Once the aerodynamic forces induced by the propellers are defined, the active torques responsible to the movement of the quadrotor can be deduced.

Movements

The upward and downward movement is the translational movement with respect to the inertial frame of the quadrotor $\overrightarrow{w}$. There is a nominal speed rotation of blades w_0 where the lift produced by the propellers compensated the weight of the quadrotor. In this case if the rotation speed if smaller than w_0 the quadrotor tends to downward, and on the contrary, it tends to upward. This movement is then related to the lift force u, composed by the sum of all rotor's lift forces:

$$u = k_\omega \sum_{i=1}^{4} \omega_i^2 \tag{7}$$

Note that, the thrust force F_d can be now write as a function of the lift force:

$$F_d = {T_{r/b}}^T \begin{pmatrix} 0 \\ 0 \\ u \end{pmatrix} = u \begin{pmatrix} \sin\varphi \sin\psi + \cos\phi \sin\theta \cos\psi \\ -\sin\varphi \cos\psi + \cos\varphi \sin\theta \sin\psi \\ \cos\varphi \cos\theta \end{pmatrix} \tag{8}$$

where $T_{r/b}$ is a rotation matrix between the local frame and the inertial frame [12] (see Fig. 3(a)), given by:

$$T_{r/b} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\pi & \sin\pi \\ 0 & \sin\pi & \cos\pi \end{pmatrix} \begin{pmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\psi \\ \cos\psi\sin\theta\sin\varphi - \sin\psi\cos\varphi & \sin\psi\sin\theta\sin\varphi + \cos\psi\cos\varphi & \cos\theta\sin\varphi \\ \cos\psi\sin\theta\cos\varphi + \sin\psi\sin\varphi & \sin\psi\sin\theta\cos\varphi - \cos\psi\sin\varphi & \cos\theta\cos\varphi \end{pmatrix} \tag{9}$$

The roll movement (τ_ϕ) is the rotation with respect to the $\overrightarrow{u}$ axis in the inertial frame resulting from the torque generated by the difference between lift forces of the left and right rotors. For instance, the movement to the left is consequence of the reduction in rotation speed of the rotors 2 and 3 (see Fig. 3(b)) of $\Delta\omega$, and inversely the growth of the rotation speed of the rotors 1 and 4, with the same $\Delta\omega$. In the same way, the pitch movement (τ_φ) is the rotation with respect to the $\overrightarrow{v}$ axis in the inertial frame. Finally, the yaw movement (τ_ψ) is the rotation with respect to the $\overrightarrow{w}$ axis in the inertial frame. Hence, the roll, pitch and yaw torques can be written as:

$$\tau_\varphi = k_a(f_1 - f_2 - f_3 + f_4) \tag{10}$$
$$\tau_\theta = k_a(f_1 + f_2 - f_3 - f_4) \tag{11}$$
$$\tau_\psi = k_c(f_1 - f_2 + f_3 - f_4) \tag{12}$$

where $k_a = l/\sqrt{2}$, with l the distance between the center of rotor i and the center of the rigid cross structure, and k_c is the report between drag and thrust coefficients, i.e. $k_c = k_d/k_\omega$.

Now, once can define the control inputs $(u, \tau^T)^T$ of our system in terms of forces of the four rotors by the relation:

$$\begin{pmatrix} u \\ \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ k_a & -k_a & -k_a & k_a \\ k_a & k_a & -k_a & -k_a \\ k_c & -k_c & k_c & -k_c \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}}_{f} = Mf \tag{13}$$

In summary, the coupled Lagrangian form of the dynamics is given as:

$$m\ddot{\xi} = u \begin{pmatrix} \sin\varphi\sin\psi + \cos\phi\sin\theta\cos\psi \\ -\sin\varphi\cos\psi + \cos\varphi\sin\theta\sin\psi \\ \cos\varphi\cos\theta \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \tag{14}$$

$$J\ddot{\eta} = \begin{pmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} \tag{15}$$

where $u \in R$ and $\tau \in R^3$ are the control inputs.

Note that this system is an under-actuated system with six outputs and four inputs. Therefore this system is also a nonlinear system. In this papers, and for low dynamics, the classical reproach with cascade loop for attitude and navigation control laws is chosen.

2.3 AR.drone attitude control

In order to achieve navigation, the attitude loop has to be tuned with low overshoot and good response time (approximatively $0.5s$). On the experimental framework, the Pixhawk card and the last firmware are used. In this context, the proposed PID controller for attitude loop is used inside the Pixhawk card. Figure 4 presents the simplified blocks of the cascade loop. A PID ($C_{pos}(z)$ in Fig. 4) is used for the each rotational angular velocity ($\dot{\theta}$, $\dot{\phi}$, $\dot{\psi}$) and a PID ($C_{speed}(z)$ in Fig. 4) is used for each angular position (θ, ϕ, ψ).

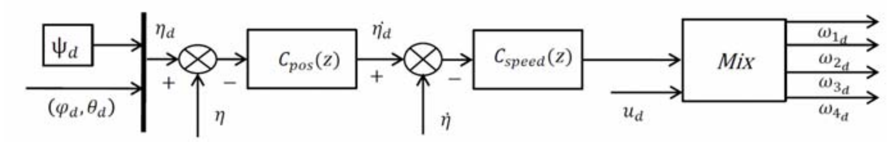


Fig. 4 Embedded Pixhawk attitude control.

The initial parameters for the PID regulators that are given in the last firmware of Pixhawk card are not well tuned for our application's need. In order to take benefit of this framework, the model has been identified collecting flight test data. So, we have obtained inertial, mass and aerodynamic coefficients for the AR.dorne, with Pixhawk card, gumstix board and USB wifi key. They are explicit shown bellow :

$I_x = I_y = 0.0039 Kg.m^2$
$I_z = 0.0077 Kg.m^2$
$m = 0.38 Kg$
$k_\omega = 9.17 \cdot 10^{-6} N/rad.s^{-1}$
$k_d = 1.37 \cdot 10^{-6} N.m/rad.s^{-1}$
$k_a = 0.13m$
$k_c = 0.15m$

The simulation was performed with a complete Matlab/Simulink model using the 6DOF block and assuming a constant inertia. Aerodynamic drag forcse induced by the movement are token into account and have been identified for low speed movements. The complete model includes coupling forces, gyroscopic effects but is not detailed in this paper.

Figure 5 shows a comparison between the experimental results and the simulated results with manual control with joystick. The joystick inputs are recorded and injected in the Simulink files. The comparison between both models is not perfect but is quite good to improve the tunning of the gains. To obtain this results, the new regulator parameters used in the Pixhawk card are now:

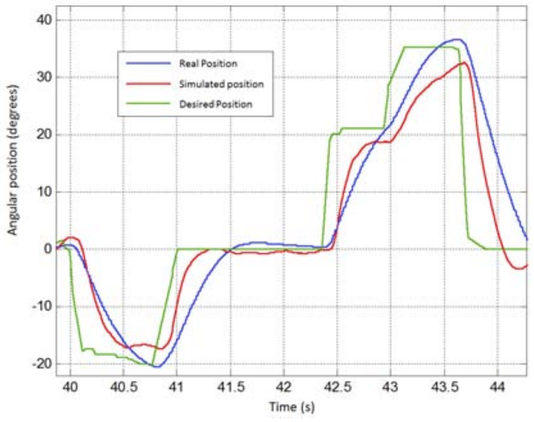


Fig. 5 Pitch attitude comparison between flight test and simulation.

$$\begin{array}{llll} roll, pitch\ attitude: & k_p = 8 & k_i = 0.1 & k_d = 0 \\ roll, pitch\ rate: & k_p = 0.13 & k_i = 0 & k_d = 0 \\ yaw\ attitude: & k_p = 0.2 & k_i = 0.15 & k_d = 0 \\ yaw\ rate: & k_p = 0.3 & k_i = 0.2 & k_d = 0 \end{array} \qquad (16)$$

The controllers outputs are mixed and multiplied by a scale factor inside the Pixhawk to deliver the rotational speed desired for each brushless motors. One can notice that, the second order equivalent model between the angular position and the command input has an overshoot of 5% and a response time around 0.6 sec. This attitude control is enough robust and smooth to be included in the navigation loop which is developed with backstepping in the next section.

3 Backstepping control

The backstepping method was developed in 90's by [8] in the aim to conceive methods for stabilizing a particular class of dynamic nonlinear systems. The principal idea is the system can be decomposed in subsystems that are themselves concentrated around an irreducible subsystem that can be stabilized. Hence, the engineer can start the control process on the level of the stable irreducible subsystem and comes back with the control to stabilize the outer subsystems.The process is finish when the outer control feedback is reached. The backstepping approach propose a recursive approach in order to stabilize the origin of a system in strict feedback control.

Usually, people use to design the backstepping control based on state variable form, which introduces unnecessary complications, as highlighted by [4]. In the same way that [4], we have chosen to apply backstepping control directly to the Lagrangian dynamics equations.

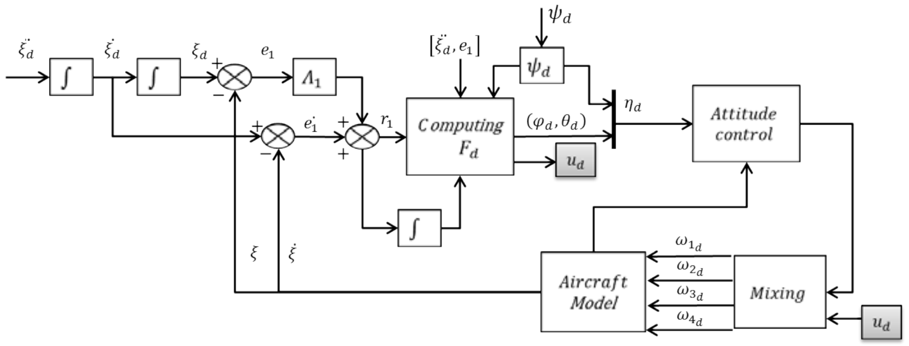


Fig. 6 Backstepping control law schema block.

In the next paragraph, the backstepping control designed is presneted, where the aim is to track a path defined as position, speed and acceleration smooth trajectories.

Position feedback control

The Figure 6 presentes the schema block used to design the backstepping control law. Next, necessary calculations to better understand this schema are detailled.

The first step to designing control law is determine the position error , and speed error dynamics. In this way, the position track error $e_1(t)$ and speed track error $\dot{e}_1(t)$ of the subsystem are described:

$$e_1 = \xi_d - \xi \tag{17}$$

$$\dot{e}_1 = v_d - v \tag{18}$$

where, ξ_d and v_d are respectively the desired positions and desired speed.

One can define also, a sliding mode error $r_1(t)$ [4] , that aggregate position error and speed error as a linear combination, also namely filtered error:

$$r_1 = \dot{e}_1 + \Lambda_1 e_1 \tag{19}$$

where, Λ_1 is a diagonal positive definite constant parameters matrix. It can be show that, as the Λ matrix is diagonal with positive entries (so, a stable system), the error e_1 is bounded as long as the filtered error r_1 remains bounded [4].

Note that, the objective is clearly to design a controller that forces the system to get r_1 small. The parameter Λ_1 is so selected for a desired sliding mode response as $e_1(t) = e_1(0)e^{-\Lambda_1 t}$, for which the point $\dot{e}_1 + \Lambda_1 e_1 = 0$ defines a stable sliding mode.

To reveal the filtered error in the dynamics equation, Eq. 19 is multiplied by the mass m of the system and afterwards derivate on time:

$$m\dot{r}_1 = m\ddot{e}_1 + m\Lambda\dot{e}_1 \tag{20}$$

$$m\dot{r}_1 = m\ddot{\xi}_d - m\ddot{\xi} + m\Lambda_1(r_1 - \Lambda_1 e_1) \tag{21}$$

$$m\dot{r}_1 = m\ddot{\xi}_d - \left(F_d - \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} \right) + m\Lambda_1 r_1 - m\Lambda_1^2 e_1 \tag{22}$$

where, F_d is given by equation 8. Rewriting Eq. 22 with a proportional K_{r_1} and an integral K_{i_1} definite positive matrix gains, we have:

$$F_d = m\ddot{\xi}_d + \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} - m\Lambda_1^2 e_1 + K_{r_1} r_1 + K_{i_1} \int_0^t r_1 dt \tag{23}$$

Note that, on can define the close-loop position error dynamics as:

$$-m\dot{r}_1 + m\Lambda_1 r_1 = K_{r_1} r_1 + K_{i_1} \int_0^t r_1 dt \tag{24}$$

$$m\dot{r}_1 = m\Lambda_1 r_1 - K_{r_1} r_1 - K_{i_1} \int_0^t r_1 dt \tag{25}$$

$$m\dot{r}_1 = -(K_{r_1} - m\Lambda_1) r_1 - K_{i_1} \int_0^t r_1 dt \tag{26}$$

F_d now help us to determine the control inputs u and τ in the design control procedure. Is this relation that shows the principal interest of the Lagrangian form to backstepping procedure.

In this sense, one can impose the yaw value ψ_d, and explicitly determine u_d, θ_d and φ_d from F_d (see Eq. 8). It can be obtained noting that $||F_d|| = u$ and writing $k_{c\psi} = \cos\psi_d$ and $k_{s\psi} = \sin\psi_d$, as:

$$F_d = u \begin{pmatrix} \sin\varphi \sin\psi + \cos\phi \sin\theta \cos\psi \\ -\sin\varphi \cos\psi + \cos\varphi \sin\theta \sin\psi \\ \cos\varphi \cos\theta \end{pmatrix} \tag{27}$$

$$F_d = ||F_d|| \underbrace{\begin{pmatrix} k_{s\psi} & k_{c\psi} & 0 \\ -k_{c\psi} & k_{s\psi} & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{C_\psi} \begin{pmatrix} \sin\varphi_d \\ \cos\varphi_d \sin\theta_d \\ \cos\varphi_d \cos\theta_d \end{pmatrix} = ||F_d||\, C_\psi \begin{pmatrix} \sin\varphi_d \\ \cos\varphi_d \sin\theta_d \\ \cos\varphi_d \cos\theta_d \end{pmatrix} \tag{28}$$

resulting in :

$$\begin{pmatrix} \sin\varphi_d \\ \cos\varphi_d \sin\theta_d \\ \cos\varphi_d \cos\theta_d \end{pmatrix} = C_\psi^{-1} \frac{F_d}{||F_d||} = \begin{pmatrix} f_{d_1} \\ f_{d_2} \\ f_{d_3} \end{pmatrix} \tag{29}$$

giving that C_ψ is a define positive rotation matrix.

As ψ_d is imposed, what allow us to always define C_ψ matrix, we have $(\varphi_d, \theta_d, u_d)^T$ computed by:

$$u_d = ||F_d||, \;\; \varphi_d = \arcsin f_{d_1}, \;\; \theta_d = \arctan \frac{f_{d_2}}{f_{d_3}} \tag{30}$$

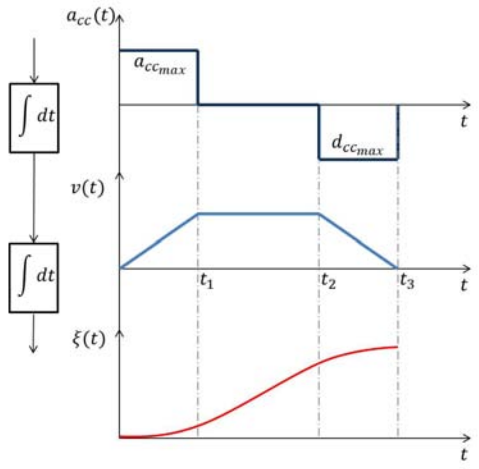


Fig. 7 Acceleration, speed and position profiles.

It is important to note that this paper does not claim the perfect control law design, neither gives a complete stability analysis. We aim to show the quality behavior of this control law once implemented in real flight testbed. This paper gives many simulation and test flights results which are presented in the section 4. Next, we present how we have proceed to generate the path reference to use as input in our application.

Automatic reference path generation

The next step was to fix some constraints related to the drone nominal flight. The most important constraint is the maximum allowed speed which we must impose to the system. Following, it can be interesting to manage the acceleration and deceleration phases with respect to the mission or simply to the desired movement. In this sense, for instance, one can chosen a trapezoidal profile for the speed reference. It is illustrate in the Figure 7.

Note that, in this example, we have three different phases on the path profile. First there is a little period, in that, with the maximum acceleration we have a growing speed. Next, with a null acceleration the speed remains constant. In the end, with a maximum deceleration we have the speed that decreases to zero. Theses three phases should be defined in function or with respect to the movement's needs. One can impose an acceleration and a deceleration symmetric periods, so $d_{cc_{max}} = -a_{cc_{max}}$.

The equations allowing us to determine the maximum acceleration and the maximum speed are the physical fundamental equations of the uniformly accelerated rectilinear movement. It allows us to determine the relationship between acceleration, speed and the position displacement. It can be shown that, in developing this fundamental equations, we can obtain the $a_{cc_{max}}$. For instance, we show the equation used to determine the acceleration on x axis:

$$a^{x}_{cc_{max}} = \frac{\Delta x}{\Delta a_{cc}(\Delta_t - 2\Delta a_{cc}) + \Delta^2_{a_{cc}}} \tag{31}$$

where Δx represents the displacement in x, Δa_{cc} the time period to acceleration phase, and Δ_t the total duration of displacement. Note that this total duration depends directly on the maximum speed of the quadrotor.

Note that this exposed calculation is given with an illustrative aim. Some results we show next demonstrate that the control law developed can be able to track any smooth path, twice derivable, with respect to the dynamics constraints of the quadrotor. The automatic path generation is given by a independent and real time system (implemented in Matlab Real Time Target) that is able to generate a new position, speed and acceleration reference profile as the desired position changes.

For experimentation purposes, two flight mode are considered: automatic and manned. When the manned mode is set, the automatic profile generation is inactivated, the function returns the current position as desired position and zero to desired speed and acceleration. As it, we can avoid reference pick, which could damage the system. If the operator chosen the automatic flight mode, references are automatically generated when the shape of the profile and the desired position are defined by the user. References are represented as a table giving the acceleration, the speed and the position inputs elements for each sample time. Once desired position is reached, the automatic generator holds the current position of the quadrotor.

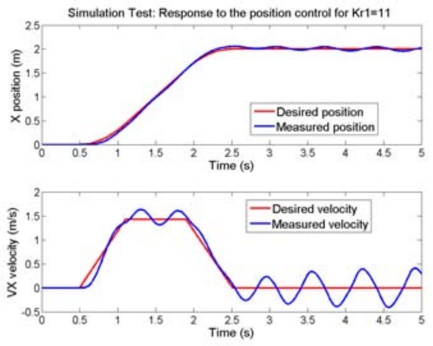
Backstepping gains regulation

We have determined the input reference, now it is mandatory to regulate control law gains to better follow these input references. To understand the role of each gains, please note Eq. 23. We can isolate each gain to better visualize their utilities:

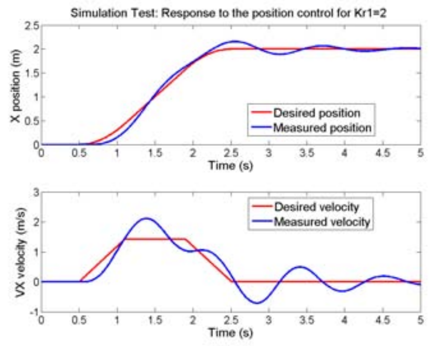
$$\begin{cases} m\Lambda_1^2 e_1 \\ K_{r_1} r_1 = K_{r_1}(\Lambda_1 e_1 + \dot{e}_1) \\ K_{i_1} \int_0^t r_1 dt = K_{i_1} \int_0^t (\Lambda_1 e_1 + \dot{e}_1) dt \end{cases} \tag{32}$$

Note that, e_1 and $\dot{e}_1$ are position and speed error, in this sense, $\Lambda_1 K_{r_1}$ and $\Lambda_1 K_{i_1}$, and, K_{r_1} and K_{i_1} are proportional and integer gains for position and for speed respectively. Therefore, Λ_1 plays a regulation role, because it balances the weight that we shaw give to position or speed error. It could be very important to switch Λ_1 with respect to the flight environment (indoor or outdoor) in which the quadrotor flights. Finally, the $m\Lambda_1$ term plays a prevention role, as this term takes into account the mass of the system. It permits to anticipate the quadrotor's payload in the positioning and to integrate its effect on the thrust force F_d.

Variating the gains values, allow us to chose the most appropriate values with respect to safety constraints. Note that, in the quadrotor case, stability is the most important one. Hence, we give priority to most conservatory values to gains. The acceleration and deceleration symmetric periods will be set as 30% of the total profile duration.



(a) Position and speed results for a strong K_{r_1}.



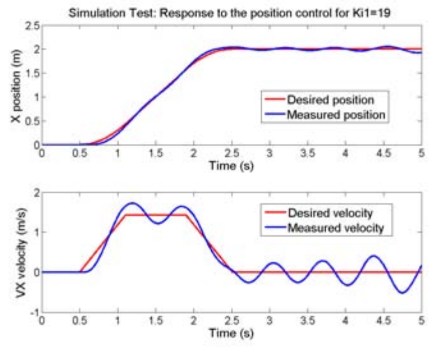
(b) Position and speed results for a weak K_{r_1}.

Fig. 8 Results concerning the K_{r_1} gain's regulation.

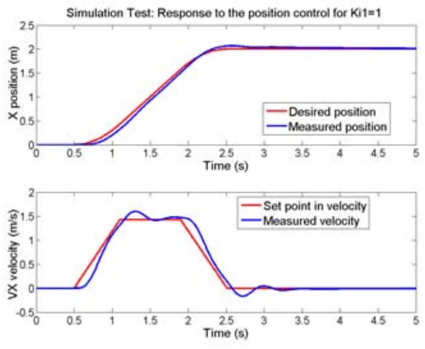
The Figure 8 shows results for Matlab simulations when we variate the gain K_{r_1}, supposing other gains constant. When K_{r_1} is to strong, the response on speed will be really fast, with a small position error. But this fat action can bring instability to the system, which will oscillate and diverge. When K_{r_1} is to weak, the response on speed will be to slow, it brings an overshoot in speed to reach the desired position really slowly.

The Figure 9 show results for Matlab simulations when we variate the gain K_{i_1}, supposing other gains constant. When K_{i_1} is to strong, the integrator forces the position regulation, and brings instability on speed, which diverges. When K_{i_1} is to weak, the system is really stable, but a little error on position is verified, which is necessary to be corrected.

Finally, the Figure 10 shows results for Matlab simulations when we variate the gain Λ_1, supposing other gains constant. When Λ_1 is to strong, we give priority to position's regulation despite of speed regulation. The error on position is smaller, but the speed regulation endure important variations, causing important oscillations.

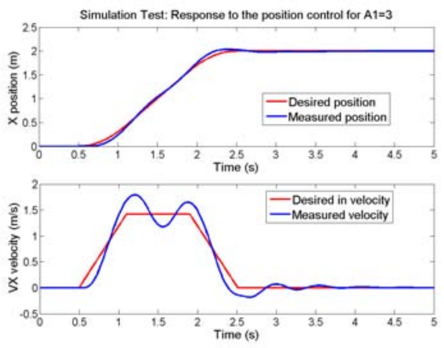


(a) Position and speed results for a strong K_{i_1}.

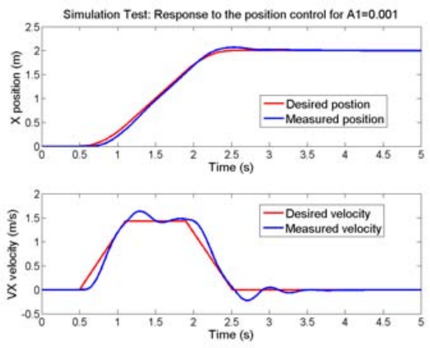


(b) Position and speed results for a weak K_{i_1}.

Fig. 9 Simulation results concerning the K_{i_1} gain's regulation.



(a) Position and speed results for a strong Λ_1.



(b) Position and speed results for a weak Λ_1.

Fig. 10 Simulation results concerning the Λ_1 gain's regulation.

When Λ_1 is to weak, the position error is small and the speed regulation has good behavior.

The conclusion of this part is that for K_{i_1} and Λ_1 gains it is preferable to have weak values. For instance, if the reference inputs changes, if the ramp reference value increases and if the time to perform displacement decreases, the system would be more solicited (bigger speed and acceleration). In this case, if we have strong gains, next to the instability, we risk to push the system effectively to the instability.

In this sense, we have fixed the gains values as $K_{r_1} = 5$, $K_{i_1} = 10$ and $\Lambda_1 = 0.6$, which we consider optimal to our application. In the Figure 11 we show the results obtained to this set of gains values. Note that the position error is small, for which, the overshoot does not pass 5%. As well, a good performance is noted for the speed, for which, we retrieve 10% of overshoot, with slightly oscillations, what does not bring prejudice to the system's stability.

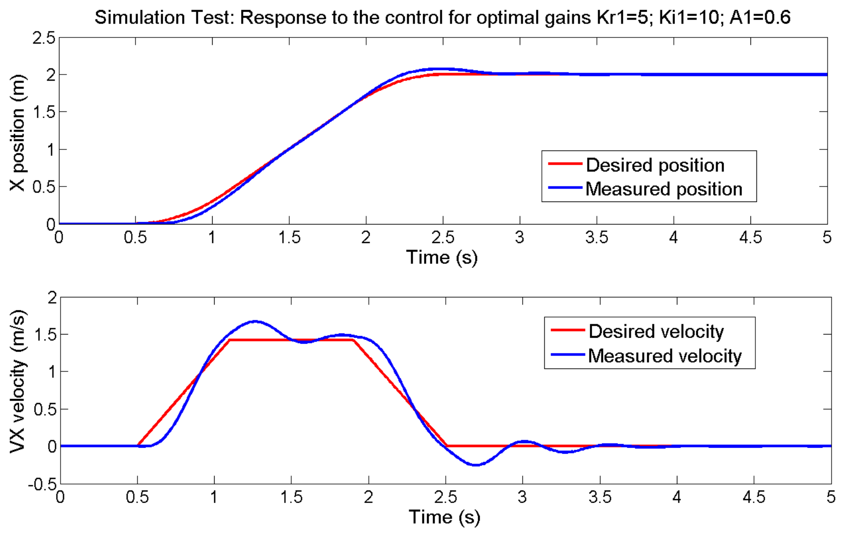


Fig. 11 Simulation results showing the chosen gains.

In our procedure, the next step, was to verify the behavior of the control when performing different smooth paths. The experimental results we show are obtained with Matlab simulation and on real flights carry out in our lab environment.

4 Experimental results

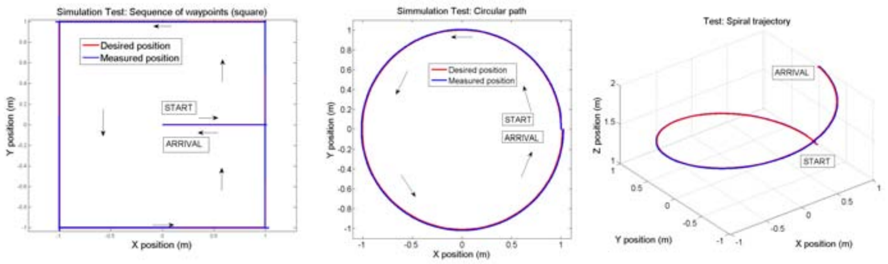
In this paper we present three cases: a sequence of waypoints designing a square, which objective is to show position error's quality; a circle, which objective is to show the adaptability of the backstepping control, hence when the speed module reference remains constant; and a spiral reference path, showing the efficiency of the backstepping control law to drive movement on the three axes (x, y, z).

Path tracking simulation

We claim that this simulation test is pertinent to validation : one can verify the sequence on automatic paths generation and the quality of the control law developed in terms of position error.

The Figure 12 shows the trajectories performed by the quadrotor in simulation for the three paths cases. Note that, the position error is really small in the three cases. The Figure 13 shows the position error. Note that this error is small for the three axis, even for z. It can be explained by the fact that even for variations on x and y, the forces computed by the control law taken into account the thrust compensation over the z axis. The backstepping control law allows to anticipate falling down on z, what can happens when a movement on x and y is required, and correct this effect before it occurs.

Once the experimental results obtained in simulation was presented, one would present the testbed environment used to real flight tests. Note that the lab platform is a rich, complete and complex framework specially designed to easy test, in real flights experimentations, the quality of control laws developed.



(a) Position results for a squared path. (b) Position results for a circular path. (c) Position results for a spiral path.

Fig. 12 Simulation tracking smooth path results.

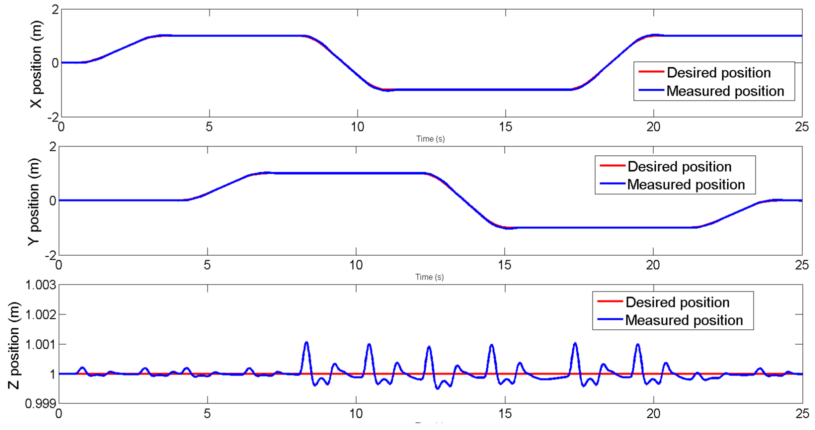


Fig. 13 Simulation results showing the sequence of target points.

Flight test results

The Figure 14 shows the results obtained in real flight using the squared path as input reference. One can note the sequence of waypoints (w.r.t. the local frame) and the evolution of the measured position and speed. The regulation is well done, even on z axis. The control law assures a small overshoot (less than 5%), and a really week error on position, which is never corrected but we can consider this error acceptable. But, on the speed tracking, overshoot appears but are not dangerous for our application and for postion tracking purpose. The altitude tracking on the z position is well done, the effect of the non linear control is seen in this case. The thrust force that has to be applied is calculated depending the attitude of the quadrotor during all the flight (taking off, cruising, landing).

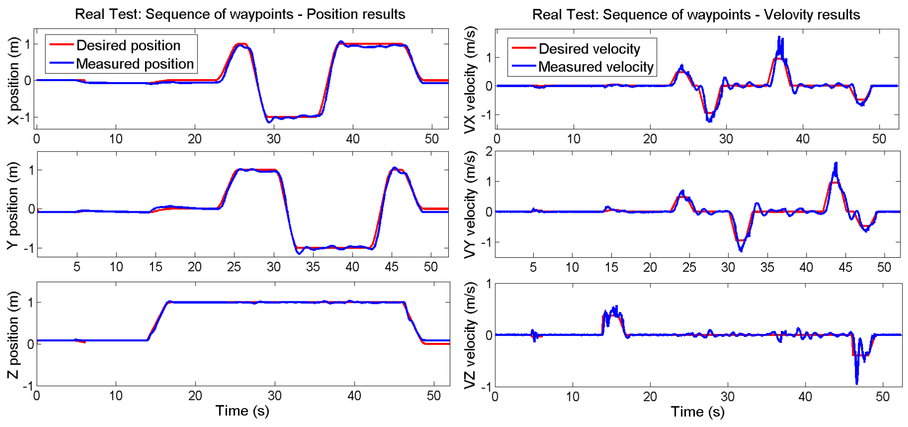
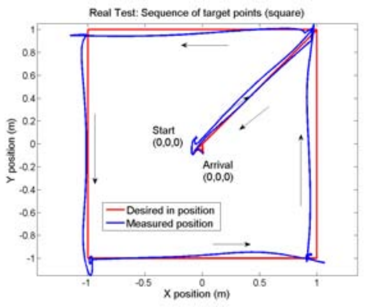
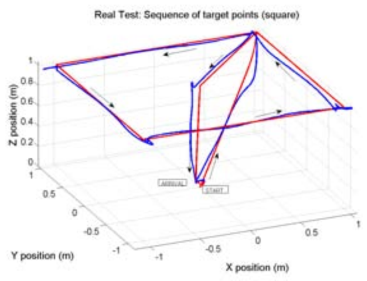


Fig. 14 Input references of position and speed, and measured ones when performing the squared path in real flight.



(a) 2D position result for a squared path.



(b) 3D position results for a squared path.

Fig. 15 Tracking smooth path flight results.

In the Figure 15 one can better visualize the trajectory performed in flight. Note that, there is a little gap on position in the x axis negative sens. This error could be explained by the fact that as the electronic used (for instance, wifi card, gumstix card, etc) are actually not placed in the center of the quadrotor structure, it displaces a little bit the center of mass. This error can be corrected modifying the F_d force in consequence. Even with this little gap, we claim that the efficacy of the control law is proved with respect to our application. Moreover, as it is shown in Eq. 8, the yaw angle is not take into account on the backstepping control law computations, but by PID control used to the attitude control in flight experimentations. In this sense, the quality observed in real flight test is strongly attached to the regulation quality of the PID attitude control.

Note that, we do not claim or compare the backstepping control law against the embedeed (comercial) ARdrone speed control law. Our aim is to define a relevant control law, that will be base for further studies, specially focused in the definition of guidance algorithms. Guidance algorithms would provide position and/or speed references in constrained environments, which justifies the chosen of the backstepping control law. As it, this first study should concretize the definition of the control law that will be implemented in our futur robotic architecture.

5 Conclusion and future work

This paper presents a real flight application of the backstepping control law to path tracking aim. For it, the testbed environment and the hypothesis related with were presented. The identification of the quadrotor's dynamic model was performed, using this testbed, in order to well regulate the control's gains (PID and backstepping) in Matlab. Hence, we do not claim any theoretical development. The principal objective of this paper is to demonstrate the applicability of such control law in real flight testbed. In this sense, we have shown that such nonlinear control law works well even if a PID law for control attitude in used in a cascade way, with respect to our application case (indoor fligth).

In the future, the developed control law will be basis to a guidance component, in which, an convenient trajectory (result of a motion planning algorithm) will be used to guide the quadrotor in a constraint environment. Use this nonlinear control law present at least two advantages for the automated trajectory planning: the range of speed that can be used is larger in this case compared to the case where a linearized approach is used; and, a good position control is ensured, which is essential when moving in constrained environments.

References

1. Bouabdallah, S., Noth, A., Siegwart, R.: Pid vs lq control techniques applied to an indoor micro quadrotor. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3, pp. 2451–2456. IEEE (2004)
2. Bouabdallah, S., Siegwart, R.: Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In: In Proceedings of IEEE Int. Conf. on Robotics and Automation, pp. 2247–2252 (2005)
3. Bouabdallah, S., Siegwart, R.: Full control of a quadrotor. In: Intelligent robots and systems, 2007. IROS 2007. IEEE/RSJ international conference on, pp. 153–158. IEEE (2007)
4. Das, A., Lewis, F., Subbarao, K.: Backstepping approach for controlling a quadrotor using lagrange form dynamics. Journal of Intelligent and Robotic Systems **56**(1-2), 127–151 (2009)
5. Das, A., Subbarao, K., Lewis, F.: Dynamic inversion with zero-dynamics stabilisation for quadrotor control. IET control theory & applications **3**(3), 303–314 (2009)
6. Doe, R.: The Orocos Project - Open RObot Control Software (2014). URL http://www.orocos.org/
7. Hamel T Mahony R, L.R.O.J.: Dynamic modelling and configuration stabilization for an x4-flyer. In: IFAC 15th triennial World congress, Barcelona, Spain, (2002)
8. Kokotovie, P.V.: The joy of feedback: nonlinear and adaptive. IEEE Control Systems Magazine **12**(3), 7–17 (1992)
9. Matthew, J., Knoebel, N., Osborne, S., Beard, R., Eldredge, A.: Adaptive backstepping control for miniature air vehicles. In: American Control Conference, 2006, pp. 6 pp.– (2006). DOI 10.1109/ACC.2006.1657678
10. Meier, L., Tanskanen, P., Heng, L., Lee, G., Fraundorfer, F., Pollefeys, M.: Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. Autonomous Robots **33**(1-2), 21–39 (2012)
11. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 2520–2525 (2011). DOI 10.1109/ICRA.2011.5980409
12. Stevens, B.L., Lewis, F.L.: Aircraft control and simulation. John Wiley & Sons (2003)
13. Vago Santana, L., Brandao, A., Sarcinelli-Filho, M., Carelli, R.: A trajectory tracking and 3d positioning controller for the ar.drone quadrotor. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, pp. 756–767 (2014). DOI 10.1109/ICUAS.2014.6842321