J Intell Robot Syst (2014) 73:387–400
DOI 10.1007/s10846-013-9953-0

A General Purpose Configurable Controller for Indoors and Outdoors GPS-Denied Navigation for Multirotor Unmanned Aerial Vehicles

Jesús Pestana · Ignacio Mellado-Bataller · Jose Luis Sanchez-Lopez · Changhong Fu · Iván F. Mondragón · Pascual Campoy

Received: 1 September 2013 / Accepted: 13 September 2013 / Published online: 5 October 2013

Abstract This research on odometry based GPS-denied navigation on multirotor Unmanned Aerial Vehicles is focused among the interactions between the odometry sensors and the navigation controller. More precisely, we present a controller architecture that allows to specify a speed specified flight envelope where the quality of the odometry measurements is guaranteed. The controller utilizes a simple point mass kinematic model, described by a set of configurable parameters, to generate a complying speed plan. For experimental testing, we have used down-facing camera optical-flow as odometry measurement. This work is a continuation of prior research to outdoors environments using an AR Drone 2.0 vehicle, as it provides reliable optical flow on a wide range of flying conditions and floor textures. Our experiments show that the architecture is realiable for outdoors flight on altitudes lower than 9 m. A prior version of our code was utilized to compete in the International Micro Air Vehicle Conference and Flight Competition IMAV 2012. The code will be released as an open-source ROS stack hosted on GitHub.

Keywords Aerial robotics · Autonomous navigation · Computer vision · Control architectures and programming · Control of UAVs · Micro aerial vehicles · Multirotor modeling and state estimation · Robotics · Unmanned aerial vehicles

J. Pestana (✉) · I. Mellado-Bataller · J. L. Sanchez-Lopez · C. Fu · P. Campoy
Centro de Automática y Robótica, CSIC-UPM, Madrid, Spain
e-mail: jesus.pestana@upm.es
URL: http://www.vision4uav.com

P. Campoy
e-mail: pascual.campoy@upm.es

I. F. Mondragón
Centro Tecnológico de Automatización Industrial, Pontificia Universidad Javeriana, Carrera 7 No 40-69, 110311, Bogotá DC, Colombia
e-mail: imondragon@javeriana.edu.co
URL: http://www.javeriana.edu.co/blogs/imondragon

1 Introduction

The motivation of this work is to study the utilization of odometry measurements as a means to stabilize the multirotor vehicle and to enable GPS-denied autonomous navigation on indoors and outdoors environments. An important difference between Vertical Take-Off and Landing (VTOL) vehicles, such as multirotors, and ground robots is that they require to be stabilized using odometry or position measurements. If these measurements have a degraded quality for a long period of time (5–30 s), there is a high risk of crashing due to small offset errors on the

Inertial Measurement Unit (IMU) estimations. For experimental testing, we have used down-facing camera optical-flow as odometry measurement which is available and implemented on a commercial multirotor, the AR Drone 2.0, as explained in [1]. This work is a continuation of prior research [2, 3] to experimental testing on outdoors environments using an AR Drone 2.0 vehicle, as it provides reliable optical flow on a wide range of flying conditions and floor textures. Optical-flow based navigation strategies are an on-going research topic on other robotics laboratories, for instance: to enable fast calculations of the optical-flow [4], for collision and obstacle avoidance [5–7], and also to enable autonomous navigation [4, 8].

The presented research is focused on multirotor platforms, that are capable of flying in cluttered areas and hover if controlled properly. In order to obtain a semi-autonomous flying platform that can be commanded with high-level commands by an operator, the state estimation and the control problem must be addressed. On the presented and recent work by our group [2, 3] a controller architecture to achieve stabilization and navigation objectives has been designed and experimentally tested on various multirotor vehicles: the AR Drone, the AR Drone 2, the Asctec Pelican and the LinkQuad quadrotors. Among other related work, the presented architecture was implemented to compete, see Fig. 1,

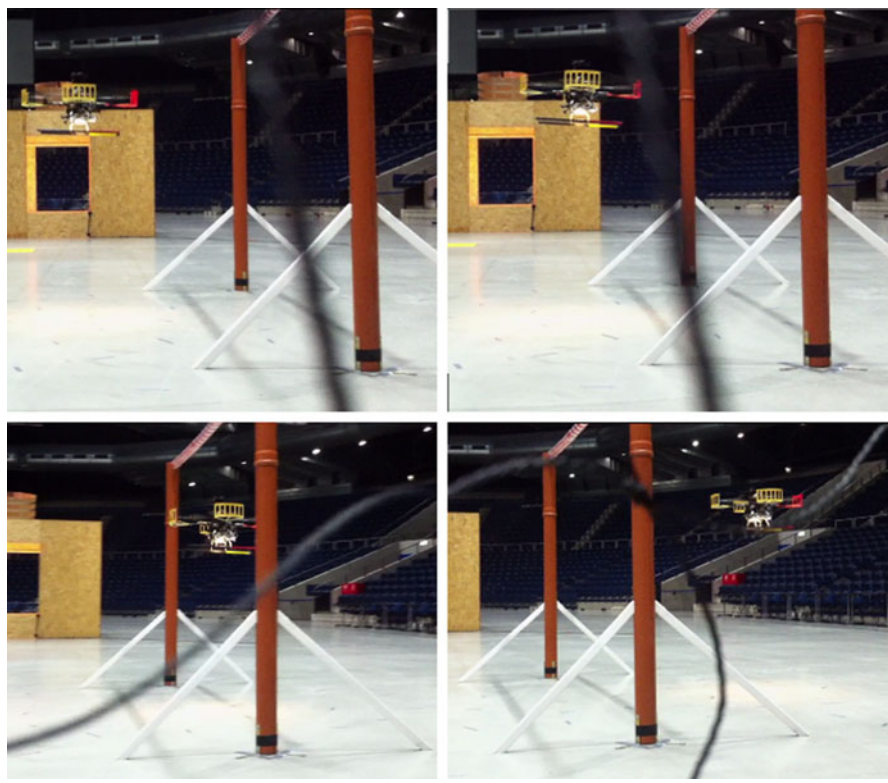

Fig. 1 The Asctec Pelican quadrotor, equipped as explained in Section 3, is shown flying in autonomous mode on the IMAV 2012 competition with an implementation of the controller presented in this paper. An EKF performs state estimation on the odometry measurements, and its estimation is fused with the position estimations from a Montecarlo Localization algorithm that is processing the Laser Range Finder readings. The quadrotor also had to be able to fly safely on areas where no obstacles were in range of the laser sensor. In this image sequence the quadrotor passes through a pair of poles using the laser sensor to estimate its position with respect to the poles. This flight gained the two IMAV 2012 awards stated in the introduction, see http://vision4uav.com/?q=IMAV12. The full video of the flight is available online on http://vision4uav.com/?q=node/323

in the International Micro Air Vehicle Conference and Flight Competition IMAV 2012, gaining two awards: the Special Award on "Best Automatic Performance—IMAV 2012" and the second overall prize in the category of "Indoor Flight Dynamics—Rotary Wing MAV". The code related to the presented work, which has been developed in our research group, will be made available in the near future as an open-source ROS Stack on GitHub [9–11], refer to Section 3.2 for more information.

2 Related Work

The following cited work has shown that the navigation control loops must be designed taking into account the non-linear dynamics of the multirotor, so that the control actions are approximately decoupled. If the control laws are well designed the multirotor can perform smooth trajectories in position coordinates, while orientating its yaw heading in any required direction.

Several labs have used a sub-millimeter accurate Vicon motion tracking system [12] to separate the control and the state estimation problems. Researchers at the GRASP lab of the University of Pennsylvania, and at the ETH—IDSC—Flying Machine Arena in the ETHZ, have been able to execute precise trajectory following [13, 14], to perform aggresive maneuvers [14, 15], and to perform collaborative tasks that require synchronization among the flying vehicles [16, 17]. Relying on Vicon motion capture systems has simplified the research problem, which has shown that state estimation is the key to enabling many autonomous applications.

On the other hand, there are research groups that have shown successful autonomous multirotor navigation capabilities using only GPS positioning data. The STARMAC project [18], from Stanford University has shown several experimental tests on outdoors ranging from trajectory tracking tasks [19] to performing flips and stall-turn maneuvers [20, 21].

Other groups have increased the situational awareness of the mUAVs using optical flow sensors and cooperative robots. For instance, a ground vehicle can estimate the position and attitude of the mUAV [22]. These two research works [5, 7] are focused on navigation in corridors and obstacle collision avoidance strategies using multiple optical flow sensors. In outdoors navigation, optical flow sensors were used in a fixed-wing mUAV in [6] to avoid the collision with trees and other obstacles in a GPS waypoint trajectory.

3 System Overview

The presented research is a continuation on previous work by the same authors [2, 3], that was implemented to participate on the IMAV 2012 indoors dynamics challenge [23]. Images from the flight that gained the awards of the IMAV 2012 competition are shown in Fig. 1, where the Asctec Pelican was controlled by the software architecture presented in this paper. In this case the localization problem was engaged using an Extended Kalman Filter (EKF) to fuse the odometry measurements, and a Particle Filter (PF) with a known map of the environment. The PF processed the Laser Range Finder readings providing estimated positions with respect to the pylons of the challenge [23]. The quadrotor also had to be able to fly safely on areas where no obstacles were in range of the laser sensor, where only the EKF provides the necessary feedback to the controller. The focus of this paper is to further researh on safe navigation based on unperfect odometry measurements, such as on-board optical flow measurements, and no position measurements are available.

This paper presents a navigation system architecture which controller can be configured depending on the limitations that are imposed by the kinematic capabilities of the vehicle, by the measurement range of the speed estimators, or by precision requirements of the task at hand. The result is an architecture that was experimentally tested on GPS-denied environments, and that can be configured depending on the requirements of each phase of a task. This allows to have a setup for fast trajectory following, and another to soften the control laws and make the vehicle navigate more precisely and slowly whenever necessary.

3.1 Hardware Architecture

The implemented hardware architecture is inspired on the AR Drone quadrotor, which capabilities are described in [1]. An Asctec Pelican quadrotor was equipped with a camera and an on-board computer to obtain a similar setup. The advantages of using the Asctec Pelican are that all the algorithms can run on the on-board computer, and that the vehicle can be equipped with more sensors such as cameras and Laser Range Finders.

The Asctec Pelican shown in Fig. 2 is equipped with an autopilot board that stabilizes the vehicle using information from GPS (only when outdoors), IMU, pressure altimeter and magnetometer fused using a Kalman Filter. This controller is embedded, closed, unmodifiable but gains are tunable. This quadrotor has also been equipped with a series of complementary sensors: a Hokuyo Scanning Laser Range Finder for horizontal depth mapping, a sonar altitude sensor for height estimation improvement and two cameras, one of them a downward looking sensor for velocity estimation based on optical flow. The optical flow is calculated using the OpenCV implementation of the pyramidal Lucas-Kanade algorithm for some image features selected by the Shi-Tomasi method. The state estimation and control processing are exeuted onboard in an Atom Board, which has a dual core Atom 1.6 GHz processor with 1 GB of RAM.

3.2 Software Architecture

The AR Drone and the Asctec Pelican were both accomodated to be compatible with the CVG Framework for interfacing with MAVs. This framework, also called "MAVwork", was developed by our research group; it is an open-source project hosted on GitHub that can be accessed in [9], and its technical details can be found on [24, 25]. One of the advantages of the framework is that it offers a common type of interface for every multirotor, allowing other software modules, such as the controller, to be compatible with multiple multirotors. The MAVwork interface has been designed to be similar to the one offered by the AR Drone, as this quadrotor has proven to be easy and reliable to setup for experimental tests. Among other advantages, MAVwork offers native take-off, landing and hovering flying modes; along with altitude hold capability. Figure 2 summarizes the interface functions that the API and drone object offer to the developer.

As explained before, the experimental tests shown on this paper have been performed using an AR Drone 2.0, as we are testing our software on outdoors environments. The advantage

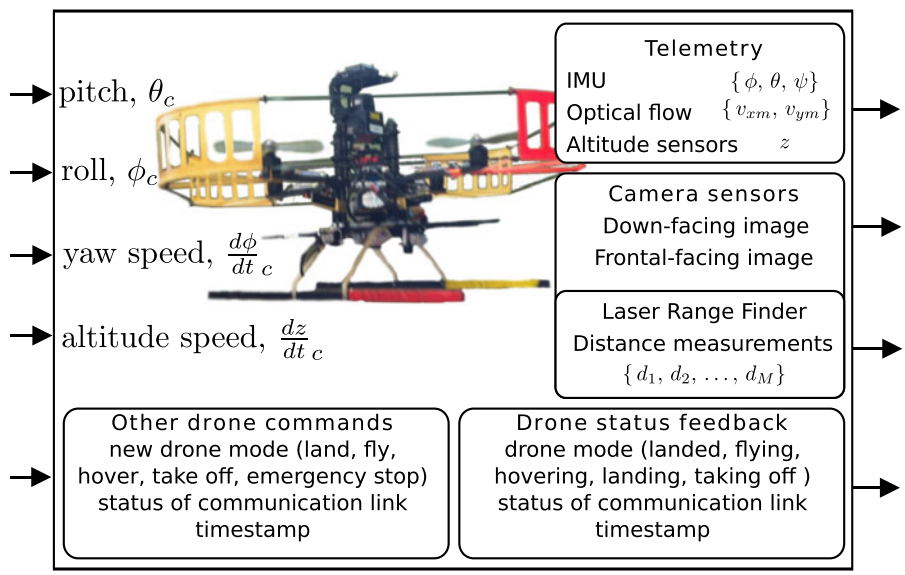

Fig. 2 Pelican on CVG Framework for interfacing with MAVs [9], user point of view. The Pelican offers several feedback channels: the telemetry package with IMU attitude, horizontal body speed and altitude data, and camera feedback channels where each camera is identified with an ID value. The drone modes are replicated from those available on the AR Drone

of MAVwork is the experiments allows us to continue improving the framework for the other multirotors as well. We perform software testing on the AR Drone 2 for multiple reasons: it is practically harmless to people and third-party objects, the high quality of its speed estiamtion, its ease of use thanks to its built-in flying modes, its low price, and its robustness to chrases.

The different components of the control and state estimation modules are shown in Fig. 3. An Extended Kalman Filter is used as data fusion algorithm to estimate the position and speed of the quadrotor. These estimates are then fed to the high-level (HL) controller that obtains the position, speed and feedforward commands that are given to the mid-level controller. The planning process that takes place in the HL controller uses a simple kinematic model of the multirotor that is specified by a set of configurable parameters. The mid-level (ML) controller calculates and sends the commands to MAVwork, which acts as interface between the controller and the quadrotor. The HL and ML controllers are further described in Section 4. The state estimation and controller algorithms are available on an open-source project called "Multirotor Controller for MAVwork" hosted on GitHub that can be accessed in [10]. Its technical details are explained in the Msc. Thesis [2] and on the present paper.

The multirotor is stabilized using odometry-based position estimates and for that reason the boundaries where the quality of these estimates is lost must be determined. For instance, the optical flow algorithm that is calculated on the down-facing images is affected by the texture and reflectiveness of the floor surface, and also by the lighting conditions of the environment. The size of the floor texture details and the computing power of the on-board computer set a maximum speed boundary on the speed estimation.

For instance, with the Pelican which tests can be found on [2, 3], the maximum speed that could be reliably measured in the indoors playground where most of the tests where carried out was about 0.5–0.75 m/s, and so, this boundary was used in our indoors tests. Based on our experience, the AR Drone 2 provides speed estimation feedback even navigating at high speeds, up to 3 m/s. The direction of the speed estimation maintains a good accuracy at all working conditions if the yaw rotations are kept low. On the other hand, the absolute value of the speed can be affected by several conditions. For example, when the drone switches from ultrasound sensor based altitude estimation to using the pressure sensor only, the speed estimation is degraded. Also the speed measurement at speed in the 2–3 m/s speed range is not accurate, but its enough to keep our controller architecture working.

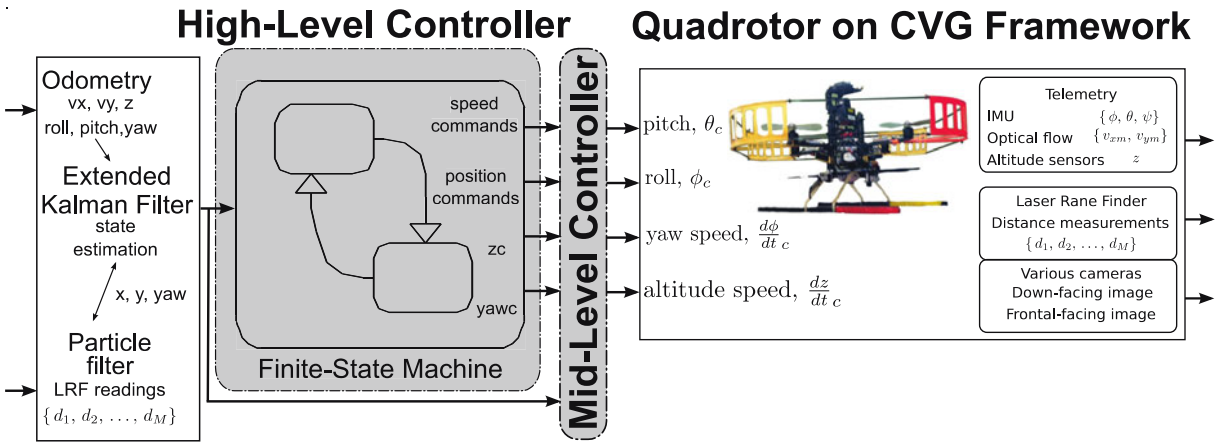

Fig. 3 General software architecture of the navigation controller. The State Estimation is used by the High-Level controller to determine optimized reachable speed commands to the mid-level controller, which sends commands to the drone through the CVG Framework [9, 24, 25]

The presented software architecture has been implemented on the AR Drone and Astec Pelican quadrotors, and will be applied in the future to other multirotor platforms, more specifically to a LinkQuad [26] quadrotor and an OktoKopter from Mikrokopter [27]. The usage of a common software architecture increases code quality shareability and enables for different research lines to work in a cohesive way towards the objectives of our group.

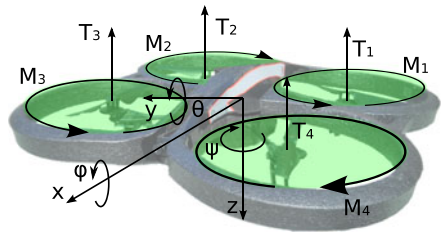

Fig. 4 Free body diagram of a quadrotor. The four propellers, and each of their performed thrust T_i and torque M_i are labeled 1–4. The euler angles are denoted $\{\phi$, roll$\}$, $\{\theta$, pitch$\}$ and $\{\psi$, yaw$\}$

4 Navigation Controller

This section introduces the navigation controller that uses a set of configurable parameters to adapt to different multirotors and sensor limitations. The parameters are inspired by the STARMAC project work described in [19]. First a simple multirotor point mass kinematic model is described. Then the high level controller state-machine is introduced with the description of the configurable parameters that are derived from the model. The final part of the section introduces the middle-level controller.

4.1 Multirotor Point Mass Kinematic Model

A multirotor is a flying vehicle that can exert force in any direction and thus can be simplified as a point mass moving in the 3D space. The vehicle tilts to exert force in the horizontal plane, and changes the average propeller speeds to control the force in the altitude coordinate. A maximum speed is also set by various reasons, for instance, due to the aerodynamic friction of the movement through the air. It is also important to take into account that the multirotor is commanded using feedback loop control laws that produce a response that can be characterized by a little set of parameters. The speed planifier can take advantage of a simple kinematic model built upon these facts in order to take decisions.

The body frame, $\{X_m, Y_m, Z_m\}$, of the vehicle, shown in Fig. 4, is described as follows. The $\{X_m, Y_m\}$ axis are horizontal and the Z_m axis points down to the floor. The X_m axis points to the front of the vehicle and the Y_m axis points to the right obtaining a orthonormal right-handed reference frame. The euler angles are denoted $\{\phi$, roll$\}$, $\{\theta$, pitch$\}$ and $\{\psi$, yaw$\}$, and they define the rotation between the body frame, $\{X_m, Y_m, Z_m\}$, and the world frame, $\{X, Y, Z\}$.

The quadrotor rigid body dynamics model, see Eq. 1 and Fig. 4, is easy to infer from physical laws and is explained in multiple articles [14, 19]:

$$
\begin{aligned}
I_x\ddot{\phi} &= \dot{\psi}\,\dot{\theta}\,(I_y - I_z) + l\,(T_1 + T_2 - T_3 - T_4)\\
I_y\ddot{\theta} &= \dot{\phi}\,\dot{\psi}\,(I_z - I_x) + l\,(T_1 + T_4 - T_2 - T_3)\\
I_z\ddot{\psi} &= \dot{\theta}\,\dot{\phi}\,(I_x - I_y) + \textstyle\sum_{i=1}^{4} M_i \qquad (1)\\
m\ddot{x} &= (-\sin\phi\,\sin\psi - \cos\phi\,\sin\theta\,\cos\psi)\\
&\quad \times \textstyle\sum_{i=1}^{4} T_i - K_{fr}\dot{x}^2\\
m\ddot{y} &= (-\cos\phi\,\sin\theta\,\sin\psi + \sin\phi\,\cos\psi)\\
&\quad \times \textstyle\sum_{i=1}^{4} T_i - K_{fr}\dot{y}^2\\
m\ddot{z} &= mg - \cos\theta\,\cos\phi\,\textstyle\sum_{i=1}^{4} T_i
\end{aligned}
$$

Where the following the following variables have been introduced: the position of the quadrotor with respect to the world frame is denoted by the $\{x, y, z\}$ coordinates, the attitude of the vehicle is represented by the yaw ψ, pitch θ and roll ϕ euler angles; the quadrotor rigid body is characterized by its mass m and its three principal mass moments of inertia $\{I_x, I_y, I_z\}$, each propeller i generates a thrust T_i and a heading torque M_i, K_{fr} is an aerodynamic friction constant and the l constant is the arm between each pair of thrusts, in this case the distance between the propellers 1 and 2, as denoted in Fig. 4.

The following control loops are usually implemented inside the autopilot board of the multirotor: the attitude (roll, ϕ, pitch, θ, and yaw, ψ) control loops and altitude, z, control loop. The autopilot boards usually accept roll, pitch, yaw speed and altitude speed references. So that the saturation bounds for these variables is set by the autopilot board capabilities. Thus, it is only required to further develop the equations of mouvement in the horizontal plane.

If the flight is performed at constant altitude then, $\sum_{i=1}^{4} T_i \approx mg$, and taking the approximation to low roll and pitch angles, then the equations that define the mouvement in the horizontal plane are derived from Eq. 1:

$$\begin{aligned} m\ddot{x} &= (-\theta \cos\psi - \phi \sin\psi)\, mg - K_{fr}\dot{x}^2 \\ m\ddot{y} &= (-\theta \sin\psi + \phi \cos\psi)\, mg - K_{fr}\dot{y}^2 \end{aligned} \tag{2}$$

Two additional attitude variables are defined $\{\phi_v$, virtual roll$\}$, $\{\theta_v$, virtual pitch$\}$, which control de position of the vehicle in the horizontal $\{X, Y\}$ plane. They are related to the actual $\{\phi, \theta\}$ through the yaw (ψ) angle as follows:

$$l\begin{bmatrix}\theta_v \\ \phi_v\end{bmatrix} = \begin{bmatrix}\cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi)\end{bmatrix}\begin{bmatrix}\theta \\ \phi\end{bmatrix} \tag{3}$$

The attitude controller ($t_r \approx$ 200–300 ms) is significantly faster than the horizontal speed loop ($t_r \approx$ 1–2 s). This fact allows to correct the coupling introduced by the yaw commands in the horizontal speed loops using the decoupling law, defined by Eq. 3. The reference frame change defined by Eq. 3 is included in the controller, so that the $\{\phi_v, \theta_v\}$ can be used for the controller design.

The dynamic model of the multirotor is much simpler when expressed using the virtual angles and a maximum value for the quadrotor acceleration saturation can be infered from Eqs. 2 and 3 as follows:

$$\begin{aligned} \ddot{x} &\approx -g\,\theta_v - (K_{fr}/m)\,\dot{x}^2 \rightarrow |\ddot{x}| \\ &\leq g\,|\theta_{v\max}| - (K_{fr}/m)\,\dot{x}^2 \end{aligned} \tag{4}$$

$$\begin{aligned} \ddot{y} &\approx g\,\phi_v - (K_{fr}/m)\,\dot{y}^2 \rightarrow |\ddot{y}| \\ &\leq g\,|\phi_{v\max}| - (K_{fr}/m)\,\dot{y}^2 \end{aligned} \tag{5}$$

The maximum horizontal speed of the vehicle, $v_{xy\max}$, is physically limited by the aerodynamic friction. Thus, the maximum quadrotor speed is expressed by Eq. 7 and the following model:

$$\left(\dot{x}^2 + \dot{y}^2\right) \leq v_{xy\max}^2 \tag{6}$$

The altitude acceleration can be modeled similarly, but the AR Drone only accepts altitude speed commands; which is the reason to constrain the altitude speed and not its acceleration:

$$|\dot{z}| \leq \frac{dz}{dt}_{c\max} \tag{7}$$

The point mass model has to take into account that the vehicle is being commanded using feedback control laws, and that the controller will not be able to push the vehicle capabilities to the physical limits stated above. Thus the following set of parameters is selected based on this fact and on the previous considerations:

- The combination of Eqs. 7 and 6 set a maximum speed on any direction.
- The speed response of the vehicle can be characterized by the response time of the speed control loop $t_{rv\max}$
- The maximum acceleration is physically limited by Eqs. 4 and 5, and can be further determined by experimental tests. But the actual accelerations obtained during trajectotry control depend on the performances of the controller and on the smoothness of the speed references set to the speed control loop. Thus, the maximum acceleration parameters in the horizontal plane are derived $\{a_{x\max}, a_{y\max}\}$.
- The trajectory around checkpoints can be approximately modeled by a sector of circumference thus requiring a centripetal acceleration that is limited by

$$a_{c\,xy} \leq g\{\theta_{v\max}, \phi_{v\max}\} = \frac{v_t^2}{R_c} \tag{8}$$

where v_t and R_c are the average speed and curve radius during the turn.

4.2 Middle-Level Controller

The mid-level controller is shown in Fig. 5 and it is inpired on previous research from other

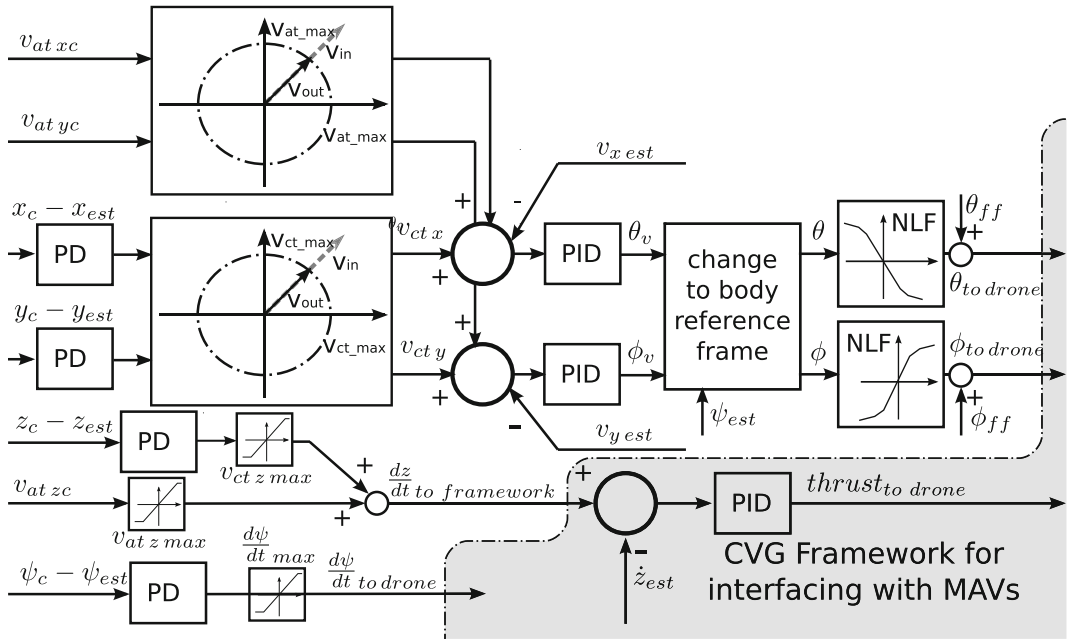

Fig. 5 The middle-level controller architecture is a cascade controller, which consists of an inner speed loop and an outer position loop. The reference change block, which implements Eq. 3 ensures that the roll and pitch commands are decoupled from the yaw heading. This allows for the yaw command to be set independently from the position and speed references. Thus, separating the left part of the diagram which is expressed in world coordinates, from the right part that is calculated in body frame coordinates

groups [13, 14, 19]. Its general architecture is a cascade controller consisting of an inner speed loop and an outer position loop. But it also includes control laws to take into account the overall system capabilities. Some characteristics of the control architecture decipted in Fig. 5 are:

- The position controller loop only commands achievable speeds to the speed controller loop, see left block in Fig. 5. The reference (feedforward) speed along the trajectory is saturated to $v_{\text{at max}}$ and $v_{\text{at }z\text{ max}}$; and the cross-track reference speed is saturated to $v_{\text{ct max}}$ and $v_{\text{ct }z\text{ max}}$. Both speed limits are set so that $v_{\text{at max}} + v_{\text{ct max}} \leq v_{\text{xy max}}$, $v_{\text{at }z\text{ max}} + v_{\text{ct }z\text{ max}} \leq v_{z\text{ max}}$.
- The planned along-track velocity $\{v_{\text{at }xc}, v_{\text{at }yc}, v_{\text{at }zc}\}$ is lower than the maximum velocity, thus, giving the relative position controller a speed margin to work on.
- The reference change block, which implements Eq. 3 ensures that the roll and pitch commands are decoupled from the yaw heading.
- The aerodynamic friction is partially linearized, in the NLF blocks, using the inverse of the identified aerodynamic friction.
- In the PID modules the derivate action is filtered to limit noise amplification and the integral action features anti-windup saturation.
- The variables $\{\theta_{ff}, \psi_{ff}\}$ are feed-forward commands that are calculated by the state machine using the planned acceleration and Eqs. 4 and 5 without considering the aerodynamic friction.

4.3 High-Level Controller - State-Machine

The HL controller is implemented using a Finite State Machine (FSM), which uses the state estimation from the EKF to calculate the mid-level controller references. The FSM has three states

corresponding to three control strategies: hover in a desired position, follow a straight segment of the trajectory and turn to head to the next trajectory segment. The work of the FSM is to navigate along a waypoint trajectory, and it can be summarized as the repetition of the following steps:

1. Follow the current straight segment, accelerating at first, and slowing down before reaching the next turn,
2. Perform the turn, controlling the speed direction to achieve a soft alignment with the next straight segment.

Every time the controller output is calculated the FSM calculates the planned speed at the current position $v_{\text{plan}}(s)$:

1. The speed, $\{V_{\text{turn}\,i}\}$, inner turn angle at checkpoint, $\{\alpha_i\}$, and radius, $\{R_{\text{turn}\,i}\}$, during each turn are precalculated each time the reference trajectory is changed, either because the whole trajectory is changed or because a waypoint was added. The radius is calculated so that the trajectory passes at a distance $R_{\text{conf}}/2$ from the checkpoint, where R_{conf} is the maximum distance at which the checkpoint is considered to be reached. The planned speed for each turn is selected either according to Eq. 8 either considering that the turn requires a stall-turn or turn-around maneuver, $v_{\text{stall turn}}$, which corresponds to a planned speed near zero.
2. The algorithm explained in Section 4.4 is used to calculate the planned speed, $v_{\text{plan}}(s)$, at the current position.

The FSM has also two additional operation modes where the quadrotor is controlled either only in position control, or in speed control. These can be used by the task scheduler to perform certain tasks.

The middle-level (ML) controller, see Fig. 3, receives position, $\{x_c,\ y_c,\ z_c\}$, yaw heading, ψ_c, speed, $\{v_{\text{at}\,xc},\ v_{\text{at}\,yc},\ v_{\text{at}\,zc}\}$, and tilt feed-forward, $\{\theta_{ff},\ \psi_{ff}\}$, commands from the FSM, as shown in Fig. 5. The presented controller, which is inspired on those presented in [14, 19], allows to use single mid-level architecture for the three control modes: trajectory, position and speed modes. These references to the ML controller are calculated by the FSM in such a way that the relative position and speed commands are orthogonal:

- The position references $[x_c,\ y_c,\ z_c]$ are caltulated projecting the estimated position $[x_{\text{est}},\ y_{\text{est}},\ z_{\text{est}}]$ onto the reference trajectory.
- The along-track speed commands, $[v_{\text{at}\,xc},\ v_{\text{at}\,yc},\ v_{\text{at}\,zc}]$, are parallel to the trajectory $v_{\text{plan}}(s)\,\mathbf{u}_{\text{tangent}}$.
- The along-track speed commands, $[v_{\text{at}\,xc},\ v_{\text{at}\,yc},\ v_{\text{at}\,zc}]$, are derivated and smoothed using a low-pass filter to obtain an acceleration command based on Eqs. 4 and 5, so that the feed-forward attitude references, $\{\theta_{ff},\ \phi_{ff}\}$, are calculated as follows:

$$\{\theta_{ff},\ \phi_{ff}\} = \left\{ -asin\left(\frac{\ddot{x}_m}{g}\right),\ asin\left(\frac{\ddot{y}_m}{g}\right)\right\}$$

4.4 Speed Planner Algorithm

The planned speed depends on the current speed and position of the multirotor, and on the previous and subsequent trajectory waypoints. The speed planner uses a uniform acceleration motion model to calculate both: the acceleration profile from the previous trajectory waypoints, and the deceleration profile to perform the following turns succesfully. In addition to this model the speed loop response is modeled taking into account its response time.

The algorithm calculates the following distances for each of the neighbouring waypoints:

1. $\Delta d_{\text{chk}\,i}$, the distance to waypoint i from the current estimated quadrotor position.
2. $\Delta d_{\text{turn}\,i} = (R_{\text{turn}\,i} + \frac{R_{\text{conf}}}{2})\,sin(\alpha_i/2)$, the distance to waypoint i when the turn is to be started; where R_{conf} is the waypoint maximum clearance distance, $\{R_{\text{turn}\,i}\}$ is the planned turn radius at waypoint i and $\{\alpha_i\}$ is the inner turn angle at waypoint i.
3. $\Delta d_{\text{trv}\,i} = t_{\text{rv max}}\sqrt{v_{x\,\text{est}}^2 + v_{y\,\text{est}}^2}$, the distance required by the speed loop to slow down, where $\{v_{x\,\text{est}},\ v_{y\,\text{est}}\}$ is the current estimated velocity and $t_{\text{rv max}}$ is the speed loop response time.

Then the optimal speed for the neighbouring checkpoints, v_{plan_i}, is calculated depending

on the sign of $\Delta d = (\Delta d_{\text{chk}\,i} - \Delta d_{\text{turn}\,i} - \Delta d_{\text{trv}\,i})$. If $\Delta d \geq 0$ then $v_{\text{plan}_i} = \sqrt{V_{\text{turn}\,i}^2 + 2a_{\text{xy max}}\,\Delta d}$, else if $\Delta d < 0$ then $v_{\text{plan}_i} = V_{\text{turn}\,i}$. Where $\{V_{\text{turn}\,i}\}$ is the planned speed at turn i. Finally the planned speed at the current position, $v_{\text{plan}}(s)$, is calculated as the minimum of the precaltulated optimal speeds $v_{\text{plan}}(s) = arg\,min_i\,(v_{\text{plan}_i})$. The coordinate s denotes the position of the quadrotor along the trajectory, and the expression $v_{\text{plan}}(s)$ highlights that the planned speed depends only on s and the following set of configurable parameters.

4.5 Configurable Parameters Set

From the prior explanation the set of configurable parameters is derived:

- $\{v_{\text{at max}},\ v_{\text{ct max}},\ v_{\text{at}\,z\,\text{max}},\ v_{\text{ct}\,z\,\text{max}}\}$, are the saturation speed limits used inside the middle level controller that are imposed by either the kinematic capabilities of the vehicle, or the measurement range of the onboard sensors or the precision requirements of the task at hand.
- $\{R_{\text{conf}},\ v_{\text{stall turn}},\ t_{\text{rv max}},\ a_{\text{xy max}}\}$, are the parameters that are used to determine the planned speed at each turn, $\{V_{\text{turn}\,i}\}$, and to calculate the planned speed, $v_{\text{plan}}(s)$, at every controller step iteration.

5 Experimental Results

This section describes five separate experimental flights that show the navigation capabilities of the presented system architecture. All the flights are trajectory tracking tests in autonomous navigation carried out in an outdoors environment, simulating a GPS-denied situation. Thus, GPS was unavailable and unsued during the tests. All the flights were performed with the AR Drone 2.0 with the outdoors hull, but similar flights with other drones can be found in [3]. The flight trajectories are shown in Figs. 6, 7 and 8. The system has been tested following regular polygon trajectories on the XY plane, with varying altitude on each waypoint, of values of 0–3 m higher than the first checkpoint's altitude. A summary of the controller performances in these tests is shown in Table 1.

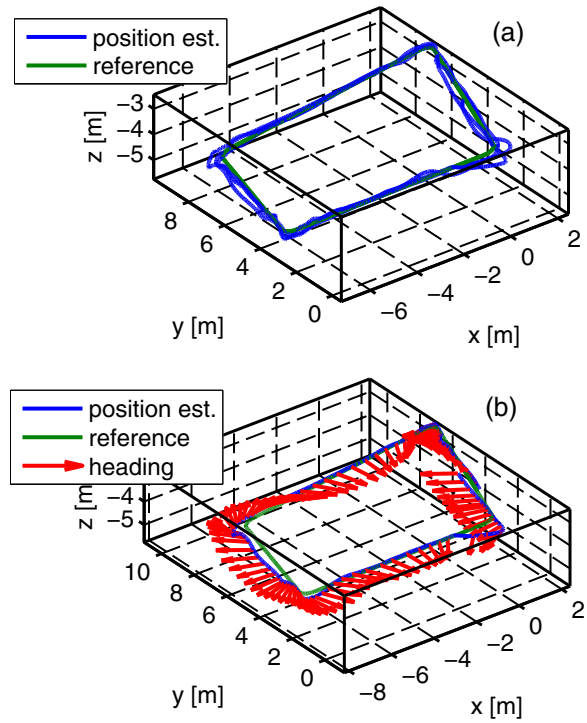

Fig. 6 Pseudo-square trajectory following experimental test on an outdoors environment performed using our achitecture and an AR Drone 2.0. The **a** graph shows the referece and followed trajectory in the XYZ space, and the **b** graph additionally shows the vehicle heading during the second lap of the experiment were the yaw controller interaction with the trajectory following task was tested. The trajectory is a square on the XY plane with varying values of altitude, on the 2.5–5.5 m range, at each checkpoint. The test consisted on three laps, where the yaw heading control was tested on the second lap, which is plotted on graph (**b**). Some performance parameters obtained from this test are shown in Table 1

The AR Drone 2.0 was configured with the following parameters values. The maximum horizontal speed was set to 2.75 m/s, where $v_{\text{at max}} = 2.0$ m/s, $v_{\text{ct max}} = 0.75$ m/s. The FSM parameters were set to $R_{\text{conf}} = 0.25$ m, $v_{\text{stall turn}} = 0.25$ m/s, $t_{\text{rv max}} = 1.20$ s, and the maximum acceleration to $a_{\text{xy max}} = 0.70$ m/s^2. The configuration parameters were selected to obtain high speeds on the straight segments, with the objective of studying the controller performance on this configuration on outdoors environments.

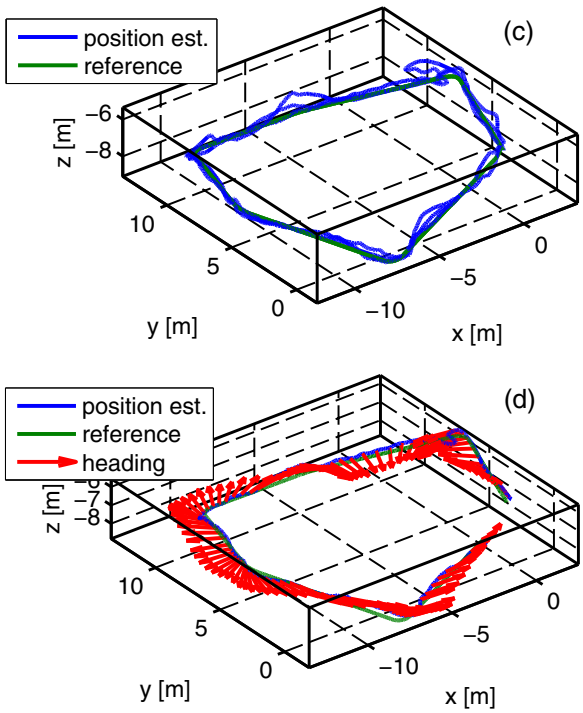

Fig. 7 Pseudo-hexagon trajectory following experimental test on an outdoors environment performed using our achitecture and an AR Drone 2.0. The **c** graph shows the referece and followed trajectory in the XYZ space, and the **d** graph additionally shows the vehicle heading during the second lap of the experiment were the yaw controller interaction with the trajectory following task was tested. The trajectory is an hexagon on the XY plane with varying values of altitude, on the 5.5–8.5 m range, at each checkpoint. The test consisted on three laps, where the yaw heading control was tested on the second lap, which is plotted on graph (**d**). Some performance parameters obtained from this test are shown in Table 1

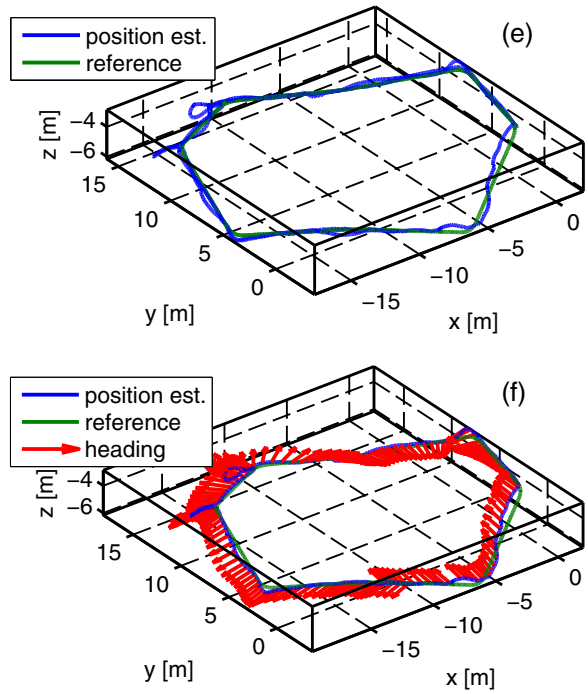

Fig. 8 Pseudo-octagon trajectory following experimental test on an outdoors environment performed using our achitecture and an AR Drone 2.0. The **e** graph shows the referece and followed trajectory in the XYZ space, and the **f** graph additionally shows the vehicle heading during the second lap of the experiment were the yaw controller interaction with the trajectory following task was tested. The trajectory is an octagon on the XY plane with varying values of altitude, on the 3.0–6.0 m range, at each checkpoint. The test consisted on two laps, where the yaw heading control was tested on the second lap, which is plotted on graph (**f**). The test was interrupted at the end of the second lap due to poor WiFi performance. Some performance parameters obtained from this test are shown in Table 1

Each experiment is shown in one figure: the first experiment consisted of a pseudo-square trajectory shown on Fig. 6, the second one was a pseudo-hexagon shown on Fig. 7, and the last one was a pseudo octagon shown on Fig. 8. As shown the drone follows a trajectory made out of straight segments and circle segments around the corners. The first graphs {(a), (c) and (e)} show only the trajectory reference and the estimated position, and the second ones {(b), (d) and (f)} show also the yaw heading during one of the laps. As can be appreciated from the graphs; the yaw control can be handled separately from the position control.

The error to the trajectory is measured against the odometry based position estimation. As expected this estimation is not accurate, and cummulates error over time. A ground truth to obtain an accurate tracking error is difficult to obtain without a motion capture system, which is the reason why it is not calculated. The drift depends on many circumstances that affect the quality of the optical flow speed estimation: texture of the floor, size of identifiable features in floor texture, illumination, altitude of the flight, yaw rotation speed, etc. To keep a good quality on the position measurements the yaw rotation speed commands

Table 1 Summary of results of the experimental tests obtained for the trajectories presented on the paper on Figs. 6, 7 and 8

Parameter units	d_{mean} [m]	d_{max} [m]	v_{mean} [m/s]	v_{max} [m/s]	ΔL_{mean} [m]	Δt_{test} [s]
Square	0.17	0.70	0.92	1.90	7.72	123.0
Hexagon	0.31	1.71	1.09	1.95	7.75	123.0
Octagon	0.32	2.90	1.17	2.10	7.75	95.0

The specified parameters are: average, d_{mean}, and maximum, d_{max}, tracking error; average, v_{mean}, and maximum, v_{max}, speed along-trajectory; ΔL_{mean} , mean distance between waypoints and Δt_{test}, duration of the test. The tested trajectories and poligons on the XY plane, with varying values of altitude on each waypoint. The tracking error is calculated from the odometry based position estimates. The mean and maximum speed are calculated taking into account the component along the trajectory, where the number of laps on each test was 2–3 laps

are saturated at 50°/s, and are usually below 30°/s. The flights were mostly anaffected during yaw heading rotation. In the flights exposed in this section the yaw rotation speed attained values of about 30–50°/s. The experimental data clarify, however, that the vehicle tends to oscillate around the reference trajectory during the yaw rotation. High yaw rotation speeds do affect the position control, but 50°/s is an acceptable value for many applications.

The altitude on each experiment was 2.5–5.5 m, 5.5–8.5 m and 3.0–6.0 m respectively. The AR Drone 2.0 can measure altitude precisely of up to 6 m [1] using the ultrasound sensor. For higher altitudes the drone has to switch to using the pressure sensor. In practice, the drone telemetry will show a good tracking on the altitude coordinate, while on reality the vehicle is correcting the altitude internally moving upwards or downwards. This was specially appreciable during some of the experiments shown on this paper, but is not visible on the plots because the sensor reading is misleading.

The selected tests show how our architecture is able to deal with moderate winds and how it takes into account automatically the curve inner angles of the trajectory at each waypoint. The trajectories have a varying number of waypoints to provide a varying inside angle of the trajectory at their waypoints. Since they are all regular polygons on the XY plane, then the higher the number of waypoints the bigger the inner angles of the trajectory will be, allowing for faster passes through them. This is appreciated by the mean speed during trajectory tracking, see v_{mean} in Table 1. The other factor that affectes the maximum and mean speeds, v_{max} and v_{mean} respectively, is the distance between waypoints; but all the trajectories where performed with similar distances. Longer distances between waypoints allow the multirotor to accelerate to and deccelerate from higher speeds.

In some occasions, the drone drifts off course in the curves of the trajectory. But this behaviour is not deterministic and the same error is not commited in every corner and is not even repeated on the next lap. However, it could be avoided lowering the $a_{xy\,max}$ and $t_{rv\,max}$ to more restrictive values. It would al be convenient to have separate values of acceleration for the curves and the straight lines, as the controller architecture does not perform equally well on straight trajectory segments than in the curves. Then the solutions are improving the controller, or adding parameters to the speed planning algorithm to take into account this issue. Another occasional behaviour is that the odometry error may increase substanially and the drone might perform a 20–50% bigger trajectory than commanded, due to odometry estimation errors; or subsequentelly cummulate a constant drift per lap on a certain direction. This behaviour might be triggered by the AR Drone's altitude sensor switching solution, but we do not have experimental evidence to ascertain the source of this occasional problem.

6 Conclusions

In this paper, we have extended the experimental work presented on a previous conference paper [3], to outdoors environments performing 3D trajectories using an additional multirotor, the AR Drone 2.0 with its outdoors hull. The tests with AR Drone 2.0 showed that this controller is able to handle wind disturbances accumulating small tracking error. However, our architecture requires a good odometry sensor which in

the presented work is directly provided by the AR Drone 2.0's hardware.

We presented our experimental work on the design of a multirotor navigation controller that allows for safer multirotor flights. The contribution of these works is two-fold: first, a discussion on various causes for multirotor navigation controllers malfunctioning is presented. Second, we derived a simple and robuts approach to address these causes natively in our middle-level controller architecture. As a further contribution to the research community, all the code related to this paper will be available on GitHub repositories.

This work is centered around the fact that the state estimation algorithms and the controller can interact in a very negative way when visual odometry estimations are faulty. This motivated our work to identify its causes and design an approach to overcome the problem. Our experimental work has led us to identify the safety boundaries under which our estimation algorithms work correctly. Then, our controller was designed to natively allow the saturation of the vehicle velocity. Although the speed can be saturated on the trajectory planning module, introducing this feature natively on the controller also allows to fly safely on speed and position control modes, or whenever the middle-level controller is utilized. The work has resulted in an overall increase of the safety and repeatibility of our experimental tests, which in turn allowed us to increase our test's efficiency and gain two awards on the IMAV 2012 competition. The experimental tests on our multirotor platforms have shown the robustness of our approach. The small trajectory tracking errors, measured against the EKF estimation, on these tests show that our efforts should be focused on improving our odometry and localization algorithms rather than the controller itself.

As an overall summary, the presented architecture was developed to allow a fast implementation on multiple multirotor vehicles. It permits to test the autonomous navigation software during real flight with a low cost vehicle, such as an AR Drone, and then easily make it portable to a more professional vehicle, such as an Asctec Pelican. The paper highlights the usage of a common drone model, and the issues that had to be solved to attain portability among multiple platforms. As future work, our group is interested on visual localization algorithms that can improve odometry and position measurements to the EKF; and the presented controller architecture will be tested in outdoors adding the GPS measurements for civilian applications research.

Acknowledgements The work reported in this article is the consecution of several research stages at the Computer Vision Group—Universidad Politécnica de Madrid. This work was supported by the Spanish Science and Technology Ministry under the grant CICYT DPI2010-20751-C02-01, and the following institutions by their scholarship grants: CSIC-JAE, CAM and the Chinese Scholarship Council. All the authors are with the Computer Vision Group, www.vision4uav.eu, which belongs to the Centre for Automation and Robotics, joint research center from the Spanish National Research Council (CSIC) and the Polytechnic University of Madrid (UPM).

References

1. The Navigation and Control Technology Inside the AR.Drone Micro UAV. Milano, Italy (2011)
2. Pestana, J.: On-board control algorithms for quadrotors and indoors navigation. Master's thesis, Universidad Politécnica de Madrid, Spain (2012)
3. Pestana, J., Mellado-Bataller, I., Fu, C., Sanchez-Lopez, J.L., Mondragon, I.F., Campoy, P.: A general purpose configurable navigation controller for micro aerial multirotor vehicles. In: 2013 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 557–564 (2013)
4. Honegger, D., Meier, L., Tanskanen, P., Pollefeys, M.: An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In: International Conference on Robotics and Automation ICRA 2013 (2013)
5. Zingg, S., Scaramuzza, D., Weiss, S., Siegwart, R.: MAV navigation through indoor corridors using optical flow. In: 2010 IEEE International Conference on Robotics and Automation (ICRA) (2010)
6. Zufferey, J.-C., Beyeler, A., Floreano, D.: Autonomous flight at low altitude with vision-based collision avoidance and GPS-based path following. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE. Online available: http://icra2010.grasp.upenn.edu/ (2010)
7. Lippiello, V., Loianno, G., Siciliano, B.: MAV indoor navigation based on a closed-form solution for absolute scale velocity estimation using optical flow and inertial data. In: 2011 50th IEEE Conference on Decision

and Control and European Control Conference (CDC-ECC), pp. 3566–3571 (2011)
8. Conroy, J., Gremillion, G., Ranganathan, B., Humbert, J.S.: Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. Auton. Robot. **27**(3), 189–198 (2009)
9. Mellado Bataller, I.: A new framework for interfacing with MAVs. https://github.com/uavster/mavwork (2012). Accessed 7 June 2013
10. Pestana, J.: A general purpose multirotor controller compatible with multiple multirotor vehicles and with the mavwork open-source project. https://github.com/jespestana/MultirotorController4mavwork (2013). Accessed 24 Apr 2013
11. A ros stack open-source implementation of the general purpose multirotor controller compatible the ar drone 1 & 2, it has been tested succesfully on the asctec pelican but integration is not provided on this version of the code. https://github.com/jespestana/ros_multirotor_navigation_controller (2013)
12. Motion capture systems from vicon. http://www.vicon.com/ (2013). Accessed 10 Sept 2013
13. Mellinger, D., Michael, N., Kumar, V.: Trajectory generation and control for precise aggressive maneuvers with quadrotors. In: Int Symposium on Experimental Robotics, (2010)
14. Michael, N., Mellinger, D., Lindsey, Q.: The GRASP multiple micro UAV testbed. IEEE Robot. Autom. Mag. **17**(3), 56–65 (2010)
15. Lupashin, S., Schollig, A., Sherback, M., D'Andrea, R.: A simple learning strategy for high-speed quadrocopter multi-flips. In: 2010 IEEE International Conference on Robotics and Automation (ICRA 2010), pp. 1642–1648 (2010)
16. Kushleyev, A., Kumar, V., Mellinger, D.: Towards a swarm of agile micro quadrotors. In: Proceedings of Robotics: Science and Systems. Sydney, Australia (2012)
17. Schölling, A., Augugliaro, F., Lupashin, S., D'Andrea, R.: Synchronizing the motion of a quadrocopter to music. In: IEEE International Conference on Robotics and Automation ICRA, pp. 3355–3360. Online available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5509755 (2010)
18. The Stanford/Berkeley Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC) project. http://hybrid.eecs.berkeley.edu/starmac/ (2013). Accessed 10 Sept 2013
19. Hoffmann, G.M., Waslander, S.L., Tomlin, C.J.: Quadrotor helicopter trajectory tracking control. In: AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, USA, August 2008
20. Hoffmann, G., Waslander, S., Tomlin, C.: Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In: 2009 IEEE International Conference on Robotics and Automation, pp. 3277–3282. IEEE (2009)
21. Gillula, J.H., Huang, H., Vitus, M.P., Tomlin, C.J.: Design of guaranteed safe maneuvers using reachable sets: autonomous quadrotor aerobatics in theory and practice. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, Anchorage, AK, May 2010, pp. 1649–1654
22. Rudol, P., Wzorek, M., Conte, G., Doherty, P.: Micro unmanned aerial vehicle visual servoing for cooperative indoor exploration. In: 2008 IEEE Conference on Aerospace (2008)
23. International micro air vehicle conference and flight competition IMAV 2012, program information for flight competition brochure. http://www.dgon-imav.org/3.0.html#c214 (2013). Accessed 10 Sept 2013
24. Mellado-Bataller, I., Mejias, L., Campoy, P., Olivares-Mendez, M.A.: Rapid prototyping framework for visual control of autonomous micro aerial vehicles. In: 12th International Conference on Intelligent Autonomous System (IAS-12), Jeju Island, Korea. Online available: http://eprints.qut.edu.au/50709/ (2012)
25. Mellado-Bataller, I., Pestana, J., Olivares-Mendez, M.A., Campoy, P., Mejias, L.: MAVwork: a framework for unified interfacing between micro aerial vehicles and visual controllers. Frontiers of Intelligent Autonomous Systems Studies in Computational Intelligence, vol. 466, pp. 165–179. Online available: http://link.springer.com/chapter/10.1007%2F978-3-642-35485-4_13 (2013)
26. UAS Technologies Sweden AB, LinkQuad quadrotor website. http://uastech.com/platforms.htm (2013). Accessed 10 Sept 2013
27. MikroKopter, OktoKopter multirotor website. http://www.mikrokopter.de/ucwiki/en/MK-Okto (2013). Accessed 10 Sept 2013