X4-MaG: a low-cost open-source micro-quadrotor and its Linux-based controller

Augustin Manecy, Nicolas Marchand, Franck Ruffier, Stéphane Viollet

X4-MaG: a low-cost open-source micro-quadrotor and its Linux-based controller

Augustin Manecy, Nicolas Marchand, Franck Ruffier and Stéphane Viollet

Reprinted from

International Journal of
Micro Air Vehicles

Volume 7 · Number 2 · June 2015

Multi-Science Publishing
ISSN 1756-8293

X4-MaG: A Low-Cost Open-Source Micro-Quadrotor and Its Linux-Based Controller

Augustin Manecy[1,2], Nicolas Marchand[2], Franck Ruffier[1] and Stéphane Viollet[1]

[1]Aix-Marseille Université, CNRS, ISM UMR 7287, Biorobotics research group, 13288, Marseille cedex 09, France
augustin.manecy@onera.fr {franck.ruffier, stephane.viollet}@univ-amu.fr
[2]GIPSA-lab laboratory, Control Systems Dept., SySCo team, CNRS-Univ. of Grenoble, ENSE3 BP 46, 38402 St Martin d'Hères Cedex, France nicolas.marchand@gipsa-lab.fr

ABSTRACT

The new open-source quadrotor platform called X4-MaG presented here was developed for academic and research applications. X4-MaG is a small, low-cost open quadrotor of only 307-grammes which offers two levels of controllers providing a manual mode and an automatic mode thanks to powerful Linux-based controller embedded onboard. The experiments presented here show the reliability of the open hardware and software embedded onboard the quadrotor. To estimate the robot s attitude, a quaternion-based complementary filter requiring very few computational resources was developed and implemented on an 8-bit Arduino board. It was also established that the stabilization feedback system based on quaternions tracks the attitude setpoints with precision up to twice greater than a classical cascaded PI controller. The controllers and estimators were designed in the Matlab/Simulink environment and directly implemented onboard the tiny Linux-based autopilot board using a custom made toolbox (RT-MaG toolbox). The autopilot was tested in the brand-new Marseilles Flying Arena with various 3-D flight trajectories and found to be highly accurate with errors of only 0.7cm in hover and less than 3.2cm at 1.2m.s^{-1} . The X4-MaG quadrotor was able to reach speeds greater than 2m.s^{-1} and reject attitude disturbances of 20' within 0.8s.

Acronyms

X4-MaG: X4 quadrotor developed in Marseille and Grenoble: an open hardware and software platform.
RT-MaG: Real-Time Marseille and Grenoble toolbox.
COM: Computer-On-Module.

1. INTRODUCTION

The last 10 years have seen the development of many quadrotor platforms, which have become popular due to their agility and the reliability of their fly-by-wire control systems. Small quadrotors are able to take off and land vertically, and their high angular speeds make them ideal candidates for performing acrobatic manoeuvres. In addition, their simple mechanical design and their robustness lend them well to research projects on autonomous aerial robots. The dynamics of quadrotors have been described in detail by ([1], [2], [3], and [4]).

The Hummingbird and Pelican by Ascending technologies are probably one of the bestknown off-the-shelf quadrotor platforms [5]. These reliable and thoroughly tested platforms are available with a large range of sensor configurations. However, these solutions are still relatively expensive and their hardware and software are not available in open access. Other configurations such as the AR drone developed by Parrot do not meet the requirements of research projects because access to the embedded sensors is too difficult and the algorithm cannot be easily tuned.

Several research groups have therefore been developing their own solutions. For example, the Grasp

Laboratory at the University of Pennsylvania have designed a small quadrotor which is able to perform very aggressive manoeuvres (up to 1800°.s^{-1} on the roll and pitch axes). Its computational resources are rather poor as it was designed to fly with the aid of a motion tracking system [6]. Another model has been proposed by [7], who developed a palm-sized quadrotor equipped with all the resources required to achieve autonomous flight based on computer vision.

Implementing a reliable functional quadrotor platform is still a tricky task because it requires a great deal of time and resources to debug and tune the complete system. Many open-source projects have emerged in order to provide solutions which can be re-used and adapted without having to start each time from scratch. For example, [8] have presented an open-source ready to-use hardware/software architecture for a large quadrotor. Another interesting project is the µAV, a palm-sized open-source quadrotor entirely designed on PCB ([9]). However, this platform does not yet provide much payload and its development is still a work in progress.

An interesting source of inspiration for developing a low cost quadrotor is the hobbyist community, which has presented many open-source platforms and software programs. For example, the Openpilot and MultiWii projects were launched by RC hobbyists ([10] and [11]). Members of the DIYdrone community have also developed the well-known Arduino-based autopilot Arducopter. As a summary, many low cost quadrotors have been described in the literature (e.g., [12], [8]) or are available on the market. However, none of them are both open-source and directly programmable with Matlab-Simulink tools. In addition, few reports are available so far in the literature on the reference tracking performances and the repeatability of the results obtained with these various open-source quadrotors.

This paper presents a new open-source and a new open-hardware quadrotor platform called X4-MaG, developed jointly by the ISM laboratory (Marseille, France) and the Gipsa-Lab (Grenoble, France), in which a particular care have been taken to assess and quantify how accurately the present small and low cost quadrotor operates in a flying arena. From the hardware point of view, the X4-MaG quadrotor is mainly composed of low cost commercial hobbyist components. From the software point of view, the Linux-based Computer-On-Module is fully supported by our new open-source Matlab/Simulink toolbox RT-MaG (see [13]), which makes the prototyping and implementation of new algorithms very fast, reliable, versatile and easy.

In addition, the present X4-MaG quadrotor operates using a fully quaternion-based open-source autopilot making the implementation more efficient. The attitude estimation was performed by a complementary filter inspired by [14] and [3], which was implemented in the present work in an optimized version using quaternions. A new version of the geometric feedback controller presented in [15] was used and enhanced by an integral action and was fully expressed with quaternions. To assess the performances of the autopilot, the quaternion-based attitude controller was compared with a classical cascaded Proportional-Integral (PI) attitude controller.

The hardware and software configuration of X4-MaG are described in the section 2. After describing the sensor fusion algorithms and the position control strategy, the flying arena in which the performances of X4-MaG were tested, is briefly described in section 3. The cascaded PI controller was compared with a quaternion-based attitude controller to show how easily the quadrotor can be used and its hovering and position tracking performances are presented in the section 4.

2. THE LINUX-BASED X4-MAG QUADROTOR

Matlab/Simulink was often used in ground station because it allows users to test and design new control algorithms very easily. [16] used Matlab/Simulink, a Vicon motion capture system and a custom-made RF module for steering a quadrotor vehicle along a trajectory. But the control algorithms are not executed onboard. Programming the X4-MaG quadrotor via Matlab/Simulink was achieved by means of our new open-source toolbox ([13]) which enable to run Simulink model in real-time on Linux based Computer-On-Module (more information are given in section 2.3.

At the hardware level, the quadrotor X4-MaG is composed entirely of low cost off-the-shelf components. Its light weight (307g) and its small frame made of a printed circuit board (PCB) make it highly resistant and robust to crashes. At the software level, the robot is equipped with two different controllers which trigger a rescue mode in case of failure of the main controller. In addition, the high level controller (the main controller) is based on the Gumstix Overo, a powerful low consumption Computer-On-Module providing the robot with huge computational resources at no cost to the payload and the endurance of the quadrotor. Since the Gumstix COM is fully supported by our new open-source Matlab/Simulink toolbox RT-MaG (see the website [13]), the robot is directly piloted via

Matlab/Simulink. Readers are invited to consult the RT-MaG website (see [13]), which gives all the information needed to assemble and program the X4-MaG quadrotor.

In this work, the quadrotor performances were evaluated using the well known Vicon s motion capture system as a localisation system (more information are available in [17]). Such a system consists of a set of calibrated infrared cameras able to locate accurately reflective markers in a 3D space (here with a sub-millimetric precision). When using multiple markers, it is possible to measure accurately both the position and the orientation of an object. More information about the precision af the Vicon's system we use are given in the appendix B.

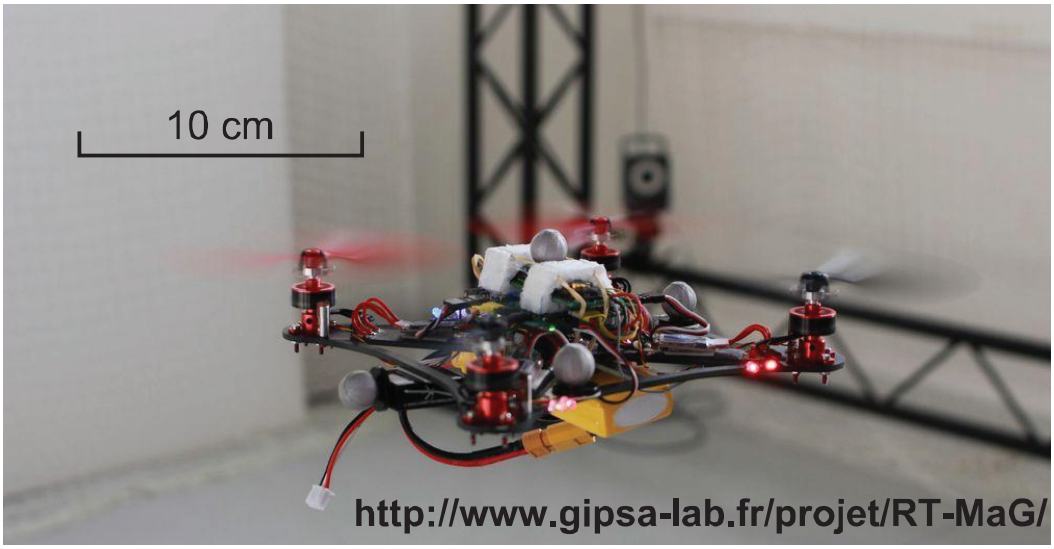

Figure 1: The X4-MaG quadrotor weighs only about 307g, and its full span is about 30cm. The complete maximum flight time is about 10 to 12 minutes without any extra load and the maximum payload is about 70g. Here, the quadrotor is equipped with marker spheres allowing to locate it accurately thanks to an external position-sensing photogrammetry system (Vicon).

2.1 The hardware

As shown in figure 1, the robot is equipped with the 20-cm span *Nanoframe* (Flyduino [18]), four "5030" propellers, four 3800KV motors, four 6A ESCs (Electronic Speed Controllers for the motors) and a 2200mAh lithium-polymer battery. The maximum flight time is about 10 to 12 minutes in normal use, the total span (including the propellers) is 30cm and the weight is about 307g including the battery pack. The X4-MAG is based on the NanoQuad frame (Flyduino), which is a small (20cm), light (60g) and genuinely crash-proof platform. This platform, which can be purchased for less than 250€, is piloted manually via a radio transmitter. It was decided to add the Gumstix Overo as a high level controller, which is a powerful Computer-On-Module (COM) providing additional computational resources. An intermediate electronic board was also added in order to finely control each rotor's speed in closed-loop and thus to obtain accurate thrust control. In this configuration, the robot, which is able to fly for less than 500€, is based on a totally open-source design. A more exhaustive list of the components, assembly instructions and open-hardware sources can be found on the website [13].

2.2 Embedded sensors and computational resources

The robot is equipped with a low-level autopilot based on a microcontroller with its 6-axis IMU (NanoWii) and a high-level autopilot based on a powerful Linux-based Computer-On-Module (Gumstix COM). The built-in NanoWii autopilot enables the robot to be piloted manually and takes over from the Gumstix if failure of the latter occurs. This architecture enabled to safely test and validate all the new algorithms while limiting the risk of crashing in the case of failure. The Gumstix COM can be programmed directly via Simulink, as it is fully supported by the RT-MaG toolbox developed at our laboratory. This considerably reduces the time elapsing before it is possible to run flight tests when developing new algorithms.

The X4-MaG quadrotor is equipped with three different electronic boards:

- The *NanoWii* [18] stabilizes the platform in the manual mode and sends sensors' values to the high-level controller in the automatic mode. This low-level controller consists of an 8-bit ATmega32u4 running at 16MHz. It is also equipped with a 6-axis IMU (the MPU6050) with three 16-bit rate gyros (with a maximum range of ±2000°.s^{-1}) and three 16-bit accelerometers (with a maximum range of ±16g). There is currently neither magnetometer, barometer nor GPS used with this quadrotor, but all of them can easily added using the NanoWii's I2C bus (e.g, the BMP085 or MS56110BA barometers and the HMC5843, HMC5883, AK8975 or MAG3110 magnetometer can be used as any I2C GPS). Currently, 4 PWM output are used to pilot the robot, and 4 additional outputs can be used to pilot until 4 other rotors (see the NanoWii user manual for more information). The manual autopilot, which is based on an open-source design, consists of a simple attitude controller receiving input reference signals from the R/C radio emitter. Attitude estimation was provided by a complementary filter described in the section 2.4 newly implemented using quaternions.
- *A Rotor Controller Board* (*RCB*) controls in closed-loop the rotational speed of each propeller. This makes it possible to accurately control the thrust of each rotor and hence, the body torques (see subsection 2.4). This controller consists of a 16-bit Microchip micro-controller (the dsPic33FJ128GP206) with a frequency of 40MHz. The thrust of each motor is controlled at 500Hz. The board provides a serial interface making it possible to easily tune the gains in each rotor's controller. All the schematics and the sources needed to build and program the rotor controller board, and the relevant documentation can be downloaded from the website [13].
- A *Gumstix* Overo AirSTORM COM is the high level controller programmed via the RT-MaG toolbox. This powerful COM features a 1GHz 32bits ARM-Cortex-A8 with 512MB of NAND memory. The Gumstix, which runs a custom-made 3.5.0 Linux patched with PREEMPT-RT, is able to run Simulink models thanks to the RT-MaG toolbox. With its breakout board (Pinto-TH), it features 4 PWM outputs (and 2 are already translated to 5.0V by the RCB), 1 SPI bus (with 2 Chip Select), 2 serial ports, 1 WIFI interface (54 MB/s), 8 GPIO and 6 10-bit ADC. The Overo AirSTORM can be easily programmed via the RT-MaG toolbox, which makes it easy to test and validate various control strategies. In addition, the board provides a wireless link (WIFI) making it really easy to monitor data and tune parameters in real time via a ground station.

The various links between the hardware components of X4-MaG are shown in figure 2. The Gumstix and the NanoWii communicate with each other via an RS232 link at 115200 Baud. The Gumstix sends the four motor input reference signals to the Nanowii, which sends the IMU's data back

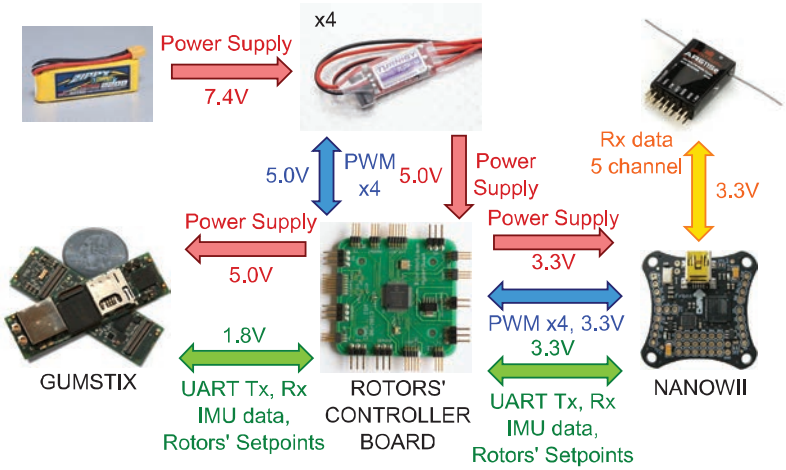

Figure 2: Hardware architecture of the quadrotor X4-MaG. The manual mode based on the Nanowii automatically takes over the control (via the minimalistic attitude estimator and controller running onboard the NanoWii) from the Gumstix autopilot if the latter crashes (i.e, the Gumstix did not sent data for more than 50ms). During the automatic mode, the Gumstix autopilot estimates the attitude (roll, pitch and yaw) and controls the 3-D position of the robot by sending to the NanoWii a setpoint corresponding to each rotor speed.

to the Gumstix. The Nanowii receives the manual command via the Radio receiver and sends the motor input reference signals to the RCB (Rotor Control Board), providing the control input signals to the rotor speed feedback loops.

Figure 3 shows the hierarchy of the various systems implemented onboard the robot. Accurate position tracking can be obtained once the robot has been linked to a motion tracking system wirelessly (see section 4). At any moment, the manual mode can be activated from the radio transmitter in order to control the robot manually.

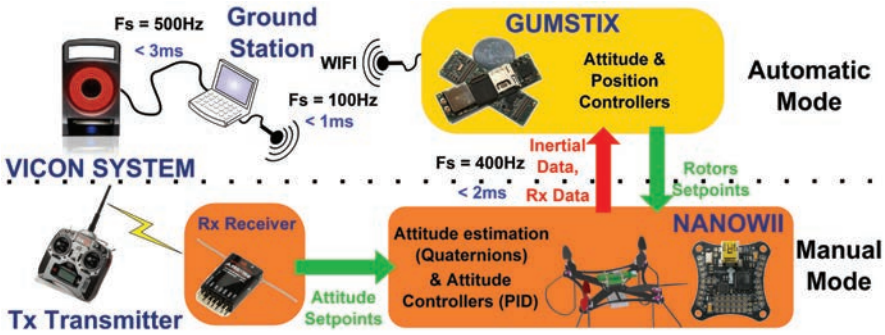

Figure 3: Interconnections between the various systems and the X4-MaG quadrotor. The sampling frequency Fs and the maximum time delays of each system are indicated. A rescue pilot automatically takes over the control of the robot and makes it hover if the Gumstix autopilot software crashes. The user can then make the robot land manually. When the Gumstix autopilot is activated, the ground station sends the position setpoints and the robot's ground truth positions via the WIFI radio link. Several parameters of the controller (gain, time constants, etc.) can be adjusted during flight from the ground station.

2.3 A Matlab/Simulink capable quadrotor

The X4-MaG quadrotor is equipped with a Gumstix Overo processor board featuring a Linux operating system patched to ensure real-time capabilities. The RT-MaG toolbox generates reliable real-time applications corresponding to a Simulink model, can run control loop at frequencies up to 1kHz and gives access to the various I/Os classically used in robotic application (UDP, SPI, I2C, RS232, PWM, GPIO and ADC). It also provides efficient debugging modes and feedback information about the real-time performances, giving users several possible metrics for optimizing their applications. This toolbox also makes it possible to perform real-time monitoring via a ground station and tune the parameters of the algorithms in real time. More complete descriptions about the RT-MaG toolbox as the sources can be found on the RT-MaG website: http://www.gipsa-lab.fr/projet/RT-MaG/ ([13]).

As a consequence, the X4-MaG quadrotor can be fully programmed via Matlab/Simulink, and all the control algorithms can be tested in simulation, in processor-in-the-loop mode and implemented on the final hardware using the same environment. This makes the development of new control strategies drastically faster.

Figure 4 gives a classic scheme for the communications between the embedded quadrotor autopilot and the ground station. The autopilot consists in a Simulink model executed in real time on a Computer-On-Module (COM). The real-time host application consists in another Simulink model executed in real time on a host computer (or a ground station). On one hand, the host computer sends high level setpoints and parameters to the embedded application. On the other hand, the host computer monitors all the signals of interest to the user.

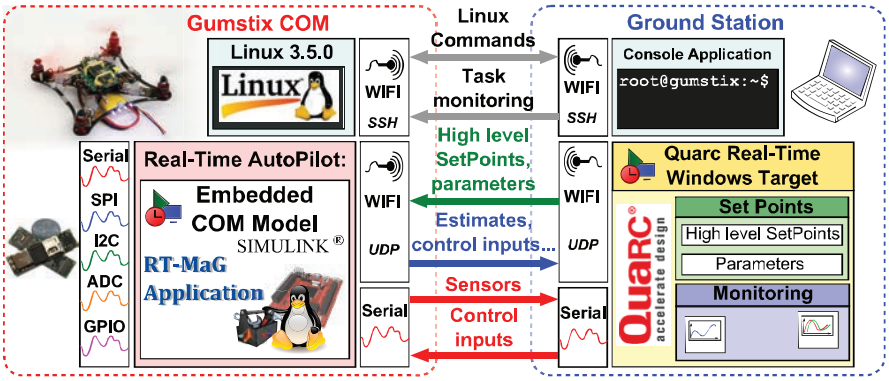

Figure 4: A classic communication scheme including the RT-MaG toolbox. The embedded Simulink model working in real-time on the Computer-On-Module uses directly the various I/Os of the quadrotor. The embedded application can be linked to a ground station via a WIFI connection in order to tune the embedded controllers or exchange data in real-time. A Processor-In-the-Loop mode is available if the COM is linked via a RS232 connection to a ground station simulating the dynamics of the X4-MaG quadrotor. Users can start or stop the real-time autopilot wirelessly (via a console) and receive continuously useful information about the embedded application (the CPU load, task execution time, the occurrence of overruns, etc.

2.4 The non-linear control strategy

A classical quadrotor model [3], which consist of a free rigid body, was used to describe the system's dynamics and expressed here using quaternion:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{v} \\ m\dot{\boldsymbol{v}} = -m\vec{\boldsymbol{g}} + \boldsymbol{F_\Sigma} \\ \dot{\boldsymbol{q}} = \dfrac{1}{2}\boldsymbol{q} \otimes \boldsymbol{p}(\boldsymbol{\Omega}) \\ \boldsymbol{I}\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times \boldsymbol{I}\boldsymbol{\Omega} + \boldsymbol{\tau_\Sigma} \end{cases} \tag{1}$$

Where $\boldsymbol{x}$ corresponds to the robot's position, $\boldsymbol{v}$ corresponds to its speed and $\boldsymbol{\Omega}$ to its rotation rate in the fixed body frame. $\otimes$ is the product of the quaternions, $\times$ is the cross product and $\boldsymbol{p}\,(\boldsymbol{\Omega})$ is a pure quaternion defined by $\boldsymbol{p}\,(\boldsymbol{\Omega}) = (0\ \Omega)^T$ (with $\boldsymbol{\Omega}$ the 3-dimensional vector of robot's rate). $\boldsymbol{F}_\Sigma$ is the sum of forces acting to the robot, and $\boldsymbol{\tau}_\Sigma$ the sum of torques. These two entities can be written:

$$\boldsymbol{F_\Sigma} = \boldsymbol{q} \odot \left(T_\Sigma.\vec{\boldsymbol{b}}_{\boldsymbol{3}}\right) + \boldsymbol{F_\Delta} = \boldsymbol{q} \odot \left(\sum_{i=0}^{4} c_T.\omega_{r_i}^2\,\vec{\boldsymbol{b}}_{\boldsymbol{3}}\right) + \boldsymbol{F_\Delta} \tag{2}$$

$$\boldsymbol{\tau_\Sigma} = \boldsymbol{\tau_\Theta} + \boldsymbol{\tau_\Delta} \tag{3}$$

where $\odot$ is the quaternion-vector product defined by $\boldsymbol{q} \odot \boldsymbol{v} = \boldsymbol{q}.\ (0\ v)^T$.\boldsymbol{q}^{-1} and $\vec{\boldsymbol{b}}_3$ is $(0\ 0\ 1)^T$ (i.e. the vertical in the fixed body frame), where c_T is the thrust coefficient (identified by static thrust tests) and ω_{ri} is the rotational speed of the rotor i. $\boldsymbol{F_\Delta}$ and $\boldsymbol{\tau_\Delta}$ are the unmodeled aero-dynamical disturbances and the unmodeled torques, respectively.

The total thrust and torques of our quadrotor (X configuration) can be expressed classically (see [3]) as follows:

$$\begin{pmatrix} T_\Sigma \\ \boldsymbol{\tau_\Theta} \end{pmatrix} = \underbrace{\begin{pmatrix} c_T & c_T & c_T & c_T \\ -l'.c_T & l'.c_T & l'.c_T & -l'.c_T \\ -l'.c_T & -l'.c_T & l'.c_T & l'.c_T \\ c_Q & -c_Q & c_Q & -c_Q \end{pmatrix}}_{\Gamma} \begin{pmatrix} \omega_{r_1}^2 \\ \omega_{r_2}^2 \\ \omega_{r_3}^2 \\ \omega_{r_4}^2 \end{pmatrix} \tag{4}$$

where T_Σ is the total thrust generated by all the rotors, $\boldsymbol{\tau_\Theta} = (\tau_\phi \ \tau_\theta \ \tau_\psi)^T$ are the body torques generated by the rotors. $l' = \frac{\sqrt{2}}{2}.l$ where l is the length of the robot's arm (the half span of the frame) and c_Q is the torque coefficient. Γ is called the thrust mixer matrix.

2.4.1 The propeller speed controller

A local closed-loop controller consisting of a simple PI controller (with anti-windup) and a feedforward term make the propellers track the rotational speed reference (which corresponds to a thrust reference set-point):

$$U_i = k_p(\omega^\star_{r_i} - \bar{\omega}_{r_i}) + k_i \int_0^t (\omega^\star_{r_i} - \bar{\omega}_{r_i})dt + U_{FF}(\omega^\star_{r_i}) \tag{5}$$

This controller yields a PWM output signal U_i from each rotor at a frequency of 500Hz (see figure 5). In the following equations, the reference values will be denoted by a star (e.g., $\omega^\star_{ri}$), the estimated values by a hat (e.g., $\hat{\boldsymbol{\Omega}}$), and the measured values by a bar (e.g., $\bar{\omega}_{ri}$). The gains k_p and k_i were adapted to each rotor in order to obtain exactly the same response to a step of rotational speed setpoint.

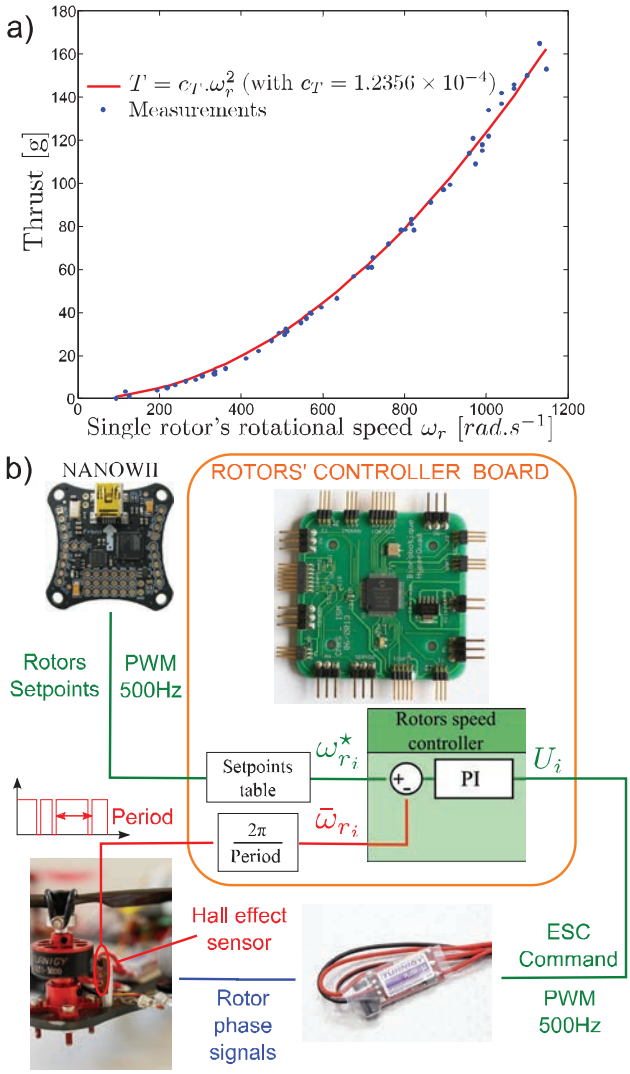

Figure 5: a) The thrust coefficient c_T was identified by applying a series of steady thrust steps. Since this identification was highly accurate, the propellers' thrust and hence the body torques can be closely controlled. b) Hardware configuration of the controller used to drive the propellers' rotational speed in closed-loop by means of a tiny Hall Effect sensor and four tiny additional magnets glued onto each motor.

2.4.2 The complementary filter

The complementary filter, which was based on that presented by [14] and [3], estimates the attitude quaternion and unbiases the rate gyros. The exponential convergence of such complementary filter was demonstrated both theoretically and experimentally in [14]. As the MPU6050 does not include any magnetometers, the yaw rate gyro's measurements were unbiased by using the yaw angle provided by the Vicon system.

$$\begin{cases} \dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\hat{\boldsymbol{q}} \otimes \boldsymbol{p}(\underbrace{\bar{\boldsymbol{\Omega}} - \hat{\boldsymbol{b}}}_{\hat{\Omega}} - \boldsymbol{\alpha}) \\ \dot{\hat{\boldsymbol{b}}} = \boldsymbol{k_b}.\boldsymbol{\alpha} \\ \boldsymbol{\alpha} = \boldsymbol{k_a} \circ \frac{\vec{\boldsymbol{g}} \times (\hat{\boldsymbol{q}} \odot \vec{\boldsymbol{g}}_{IMU})}{\|\vec{\boldsymbol{g}}\|^2} + \boldsymbol{k_v} \circ \tilde{s}\tilde{\boldsymbol{v}} \end{cases} \tag{6}$$

where $\circ$ is the Hadamard product for vectors (the elementwise vector multiplication), $\tilde{s}$ and $\tilde{\boldsymbol{v}}$ are the scalar part and the vector part, respectively, of the error quaternion between estimated quaternion and the Vicon quaternion defined by $\tilde{\boldsymbol{q}} = \bar{\boldsymbol{q}}^{-1}_{Vicon} \otimes \hat{\boldsymbol{q}} = (\tilde{s}\ \tilde{\boldsymbol{v}}\ )^T$ with $\bar{\boldsymbol{q}}_{Vicon}$ the quaternion measured by the Vicon system. Here $\boldsymbol{k_a}\} = (k_{a1}\ k_{a2}\ 0)$ with $k_{a1} > 0$ and $k_{a2} > 0$, and $\boldsymbol{k_v} = (0\ 0\ k_{v3})$ with $k_{v3} > 0$.

2.4.3 The attitude controller

Two possible means of designing the attitude controller were tested: the first one consisted of two cascaded PI controllers controlling the body rate and its attitude. The second controller consisted of a geometric controller originally designed on SO(3) by [15] and newly implemented here with the quaternions. These attitude controllers receive attitude setpoints $\boldsymbol{\Theta}^\star = (\phi^\star\ \theta^\star\ \psi^\star)^T$ from the position controller described below, and compute the three body torques $\tau^\star$ to generate.

2.4.4 Cascaded PI attitude controller

Here, the attitude was stabilized by using two levels of cascaded PI controllers. The first one controls the angular speed of the robot by providing the following torque setpoints:

$$\boldsymbol{\tau}^\star_{\boldsymbol{\Theta}} = \underbrace{\boldsymbol{k_p}(\boldsymbol{\Omega}^\star_{\boldsymbol{f}} - \hat{\boldsymbol{\Omega}}) + \boldsymbol{k_i}\int_0^t (\boldsymbol{\Omega}^\star_{\boldsymbol{f}} - \hat{\boldsymbol{\Omega}})dt}_{\tau^\star_{\Theta\,feedback}} + \underbrace{\hat{\boldsymbol{\Omega}} \times \boldsymbol{I}\hat{\boldsymbol{\Omega}} + \boldsymbol{I}\frac{d\boldsymbol{\Omega}^\star_{\boldsymbol{f}}}{dt}}_{\tau^\star_{\Theta\,feedforward}} \tag{7}$$

where $\boldsymbol{\tau}^\star_{\boldsymbol{\Theta}}$ is a 3-dimensional vector corresponding to the body torques, $\hat{\boldsymbol{\Omega}} = \bar{\boldsymbol{\Omega}} - \hat{\boldsymbol{b}}$ is the body rate vector corresponding to the unbiased gyros' measurements and $\boldsymbol{I}$ is the inertia matrix. $\boldsymbol{k_p}$ and $\boldsymbol{k_i}$ are diagonal matrices of positive gains. The two feedforward terms of $\tau^\star_{\Theta\ feedforward}$ compensate for the gyroscopic effect, and generate the torque needed to achieve the desired body rate setpoints $\boldsymbol{\Omega}^\star_{\boldsymbol{f}}$ These setpoints are obtained by filtering the setpoints $\boldsymbol{\Omega}^\star$ yielded by the angles' controller:

$$\boldsymbol{\Omega}^\star_{\boldsymbol{f}}(s) = \frac{\boldsymbol{\Omega}^\star(s)}{(\tau_{f_\Theta} s + 1)} \tag{8}$$

where $\tau_{f\Theta}$ is the time constant of the first order filter which ensure that the first derivative of $\boldsymbol{\Omega}^\star_{\boldsymbol{f}}$ is continuous (used in equation (7)). The rotational speed setpoints $\boldsymbol{\Omega}^\star$ comes from a second controller which controls thereby the attitude:

$$\boldsymbol{\Omega}^\star = \boldsymbol{k_p}(\boldsymbol{\Theta}^\star - \hat{\boldsymbol{\Theta}}) + \boldsymbol{k_i}\int_0^t (\boldsymbol{\Theta}^\star - \hat{\boldsymbol{\Theta}}) \tag{9}$$

where $\boldsymbol{k_p}$ and $\boldsymbol{k_i}$ are diagonal matrices of positive gains, and $\hat{\boldsymbol{\Theta}} = (\hat{\phi}\ \ \hat{\theta}\ \ \hat{\psi})^T$ is the vector of the Euler angles corresponding to the estimated quaternion $\hat{\boldsymbol{q}}$. $\boldsymbol{\Theta}^\star = (\phi^\star\ \theta^\star\ \psi^\star)^T$ where $\phi^\star$ and $\theta^\star$ are provided by the position controller and $\psi^\star$ is a setpoint specified by the user.

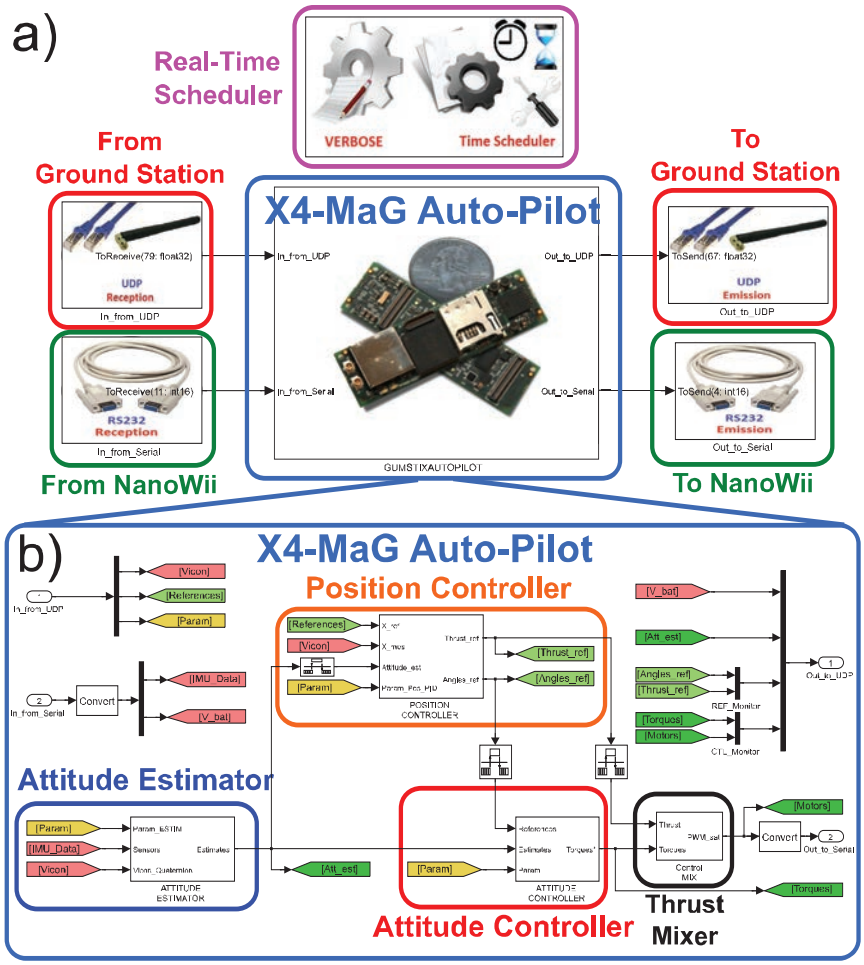

Figure 6: The simulink model of the X4-MaG auto-pilot performing the attitude estimation, position tracking and attitude stabilization. a) The I/Os of the Gumstix are easily accessible via the RT-MaG toolbox. Here, the autopilot receives high level setpoints, Vicon measurements and parameters from the ground station, and IMU data from the NanoWii. The Gumstix sends the motor commands to the Nanowii and sends the estimated data, the control inputs, etc. to the ground station. b) The details of the different subsystems of the X4-MaG autopilot.

2.4.5 Quaternion based attitude controller

The second controller is the quaternion version of the controller described by [15], to which an integral action was added. It is worth to note that the globally exponential stability of the controller described by [15] hold only under restrictive conditions (e.g., the inertia matrix is perfectly known, motor dynamics are insignificant, etc.). Theses assumptions cannot be guarantee experimentally, however such a controller (in its rotation matrix formulation) was already proven to be really efficient experimentally ([19]). And this work shows that the quaternion version of these controller was able of good tracking performances on a real quadrotor (see section 4.2).

Assuming the attitude setpoints delivered by the position controller to be $\mathbf{\Theta}^\star(t) = (\phi^\star(t)\ \theta^\star(t)\ \psi^\star(t))^T$, a new attitude setpoint $\mathbf{\Theta}_f^\star(t)$ is defined, corresponding to $\mathbf{\Theta}^\star(t)$ filtered at order 3 to ensure that its first and second derivative $\mathbf{\Omega}_f^\star$ and $\dot{\mathbf{\Omega}}_f^\star$ are continuous.

Finally the quaternion setpoint $\boldsymbol{q}_f^\star$ is extracted from $\mathbf{\Theta}_f^\star(t)$ and the attitude tracking error vector $\boldsymbol{e}_q$ (3×1 vector) can be defined as follows:

$$\boldsymbol{e_q} = \tilde{s}_{e_q}.\tilde{\boldsymbol{v}}_{\boldsymbol{e_q}} \tag{10}$$

where $\tilde{\boldsymbol{s}}_{\boldsymbol{eq}}$ and $\tilde{\boldsymbol{v}}_{\boldsymbol{eq}}$ are the scalar part and the vector part of the tracking quaternion error defined by $\tilde{\boldsymbol{q}}_{\boldsymbol{eq}} = \hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q}^\star$.

The body rate tracking error is written:

$$\boldsymbol{e_\Omega} = \left[(\hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q_f}) \odot \boldsymbol{\Omega_f^\star}\right] - \hat{\boldsymbol{\Omega}} \tag{11}$$

Then, the attitude controller designed on SO(3) which yields the following body torques can be written as:

$$\begin{aligned}\boldsymbol{\tau_\Theta^\star} = \boldsymbol{I}\left(\boldsymbol{k_R e_q} + \boldsymbol{k_I}\int_0^t \boldsymbol{e_q} dt + \boldsymbol{k_\Omega e_\Omega}\right) \\ + \hat{\boldsymbol{\Omega}} \times \boldsymbol{I}\hat{\boldsymbol{\Omega}} - \boldsymbol{I}\left(\hat{\boldsymbol{\Omega}}_\times(\tilde{\boldsymbol{q}}_{\boldsymbol{e_q}} \odot \boldsymbol{\Omega_f^\star}) - \tilde{\boldsymbol{q}}_{\boldsymbol{e_q}} \odot \dot{\boldsymbol{\Omega}}_{\boldsymbol{f}}^\star\right)\end{aligned} \tag{12}$$

2.4.6 The 3D position controller

This controller is an updated version of a classical position controller ([20]). It adjusts the roll angle ϕ and the pitch angle θ in order to track the linear position setpoint $x^\star_1$ and $x^\star_2$. All in all, it gives:

$$\begin{cases}\phi^\star = sin^{-1}\left(\dfrac{m}{T_\Sigma^\star}\left(\ddot{x}_{f1}^\star sin\hat{\psi} - \ddot{x}_{f2}^\star cos\hat{\psi}\right)\right) \\ \theta^\star = sin^{-1}\left(\dfrac{m}{T_\Sigma^\star}\left(\ddot{x}_{f1}^\star cos\hat{\psi} + \ddot{x}_{f2}^\star sin\hat{\psi}\right)\right)\end{cases} \tag{13}$$

where $\ddot{x}^\star_{f1}$ and $\ddot{x}^\star_{f2}$ are the linear acceleration setpoints composed of a feedforward term and a feedback term (computed by a PID controller) as follows:

$$\ddot{x}_{fi}^\star = \underbrace{k_P\tilde{x}_i + k_I\int_0^t \tilde{x}_i dt + k_D\left(\frac{dx_{fi}^\star}{dt} - \bar{v}_i\right)}_{\ddot{X}^\star_{i_{feedback}}} + \underbrace{\frac{d^2x_{fi}^\star}{dt^2}}_{\ddot{x}^\star_{i_{feedforward}}} \tag{14}$$

where $\tilde{x}_i$ is the 3-D position error defined by $\tilde{x}_i = (x^\star_{fi} - \bar{x}_i)$, $\bar{x}_i$ and $\bar{v}_i$ are the robot's linear position and the speed measured by the Vicon system, respectively, where $x^\star_{fi}$ corresponds to the filtered setpoint $x^\star_i$, which ensures that the signals are C^3 (i.e., the feedforward term is continuous and is therefore accessible to the robot):

$$x_{fi}^\star(s) = \frac{x_i^\star(s)}{(\tau_{f_x}s + 1)^3} \tag{15}$$

The position setpoint along the vertical axis $x^\star_3$ is tracked by adjusting the total thrust T_Σ defined by:

$$T_\Sigma^\star = \underbrace{\frac{k_P\tilde{x}_3 + k_I\int_0^t \tilde{x}_3 dt + k_D\left(\frac{dx_{f3}^\star}{dt} - \bar{v}_3\right)}{cos\hat{\phi}cos\hat{\theta}}}_{\ddot{x}_{3_{feedback}}} + \underbrace{\frac{mg}{cos\hat{\phi}cos\hat{\theta}}}_{\ddot{x}_{3_{feedforward}}} \tag{16}$$

Finally, each rotor's rotational speed setpoint, corresponding to the control input vector $(T^\star_\Sigma\ \boldsymbol{\tau^\star_\Sigma})^T$, was obtained by inverting equation (4).

As Γ is a constant matrix, its inversion can be calculated off-board only once.

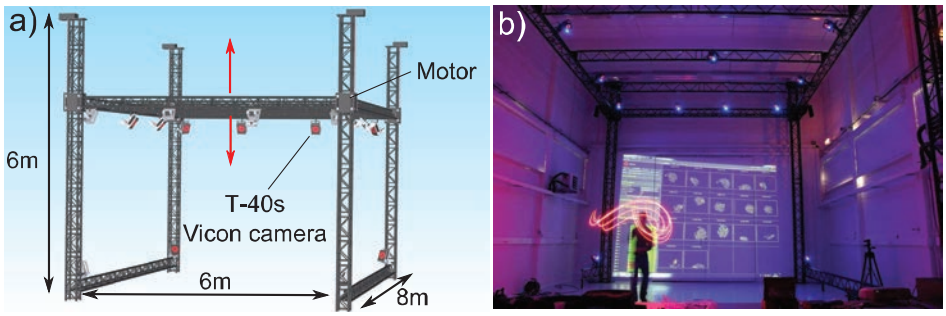

Figure 7: a) CAD of the brand new flying arena (Institute of Movement Science, Marseille, France, www.flying-arena.eu) composed of a motorized metallic tubular structure equipped with 17 T-40s Vicon cameras. b) The $6 \times 8 \times 6m^3$ flying arena during the calibration.

3. THE FLYING ARENA AND EXPERIMENTAL CONDITIONS

All the experiments presented in this paper were carried out in the brand-new Flying Arena of Marseille (see figure 7, www.marseilles-flying-arena.eu). The latter consists of a reconfigurable flight space of $6m$-wide$\times 6m$-high$\times 8m$-long covered by 17 T-40s Vicon cameras. The Computed Aided Design (CAD) of the structure is shown in figure 7a. The flight space can be reconfigured by moving the horizontal motorized structure supporting the cameras up or down. The Flying Arena is shown in figure 7b during a calibration process. In this flying arena, one or several robots can be accurately located in its largest flight space with a submillimeter precision. Once calibrated, the system provides position measurements with a 0.2-mm precision in terms of mean standard deviation (see appendix B).

3.1 Systems' interconnections

During the experiments, three different systems are interconnected:

- **A Vicon localization system**, which locates in 3D our robot at 500Hz with small latencies ($<$3ms due to the Vicon motion capture system and the network).
- **A ground station**, which sends the 3-D robot's positions measured by the Vicon system to the high-level airborne controller. This ground station monitors the computational results in real time (via WIFI) and also sends the position setpoints to the robot.
- **The X4-MaG quadrotor** equipped with a Gumstix COM and programmed via the RT-MaG toolbox. The robot receives its 3-D position and the position setpoints from the ground station, and then computes its trajectory autonomously.

The communications and the programs running on the ground station and the robot are both managed via Matlab/Simulink. During the experiments, the ground station monitors (in the Simulink environment) all the robot's data via a WIFI connection and sends the position setpoints and the Vicon measurements to the robot at a frequency of 100Hz. The Simulink model of the ground station runs in real time thanks to the Quanser's software QUARC® ([21]). The robot runs the Simulink model described by the figure 6 in real time via the RT-MaG toolbox [13]. All the computations are performed onboard by the Gumstix, i.e.: the position control loop operates at 100Hz and the attitude control loop operates at 400Hz. For safety reasons, a second autopilot is activated simultaneously on the NanoWii (the low level controller): it automatically takes over the control of the robot if the Gumstix autopilot happens to crash. If the Gumstix crashes (for whatever reason) the communication between the NanoWii and the Gumstix will be broken, and the NanoWii takes over the control if nothing was received from the Gumstix for more than 50ms. Then the Nanowii makes the robot hover, letting time to the user to takes manual control.

4. EXPERIMENTAL RESULTS

In this part, some results obtained with X4-MaG in the brand new Marseilles Flying Arena are presented.

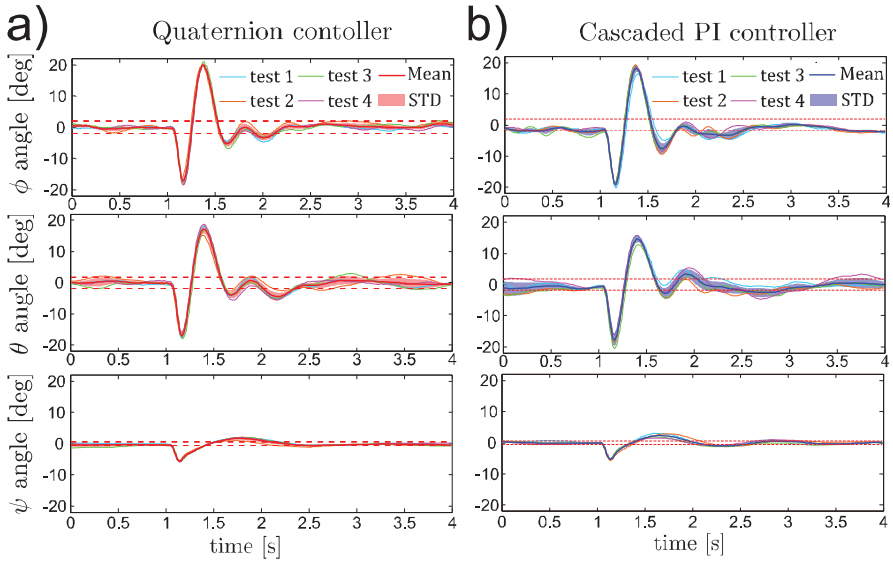

Figure 8: a) Disturbance rejection achieved by the quaternion-based attitude controller. Each test is plotted in a different color and the mean response is given by the red line. The shaded red area corresponds to the standard deviation of the various trajectories. b) Disturbance rejection achieved by the cascaded PI controllers. Each test is plotted in a different color and the mean response given by the blue line. The shaded blue area corresponds to the standard deviation of the various trajectories.

4.1 Comparison between two attitude controllers

Here, the disturbance rejection performances of the two attitude controllers presented in equations (7) and (12) are compared. Each controller was tested with the same disturbances repeated 5 times at each angle (roll, pitch and yaw). Figures 8a and 8b show that the rejection was highly reproducible in the case of both controllers (PI and quaternion). On the roll axis, 80% of the perturbation was rejected within 0.7s by the quaternion controller with a maximum value of 20°, whereas the PI controller rejected the same perturbation with a maximum value of 19° within 0.75s.

A sequence of position trajectories was then imposed on the robot and the tracking error in the attitude was measured during these manoeuvres. After a hovering phase of 15 seconds, the robot's trajectory was disturbed around the roll, pitch and yaw axes and then performed three 1-m lateral steps on the X and Y axes, followed by a 45-° tilted circular trajectory at $1 m.s^{-1}$ before landing automatically. This scenario was repeated 4 times with each of the controllers and the results were plotted as shown in figure 9. The geometric reference tracking control based on quaternions yielded a better attitude tracking results than the cascaded PI controller. The standard deviation amounted to only 2.67° and 2.15° on the roll and pitch axes, respectively, with the quaternion-based controller and 3.05° and 4.14° with the cascaded PI controller. The maximum tracking error recorded during the circular trajectory was about 20° with the cascaded PI controller but only 8° with the quaternion-based controller.

4.2 Precision and performances

This section deals with the precision of the X4-MaG's performances in the Flying Arena by implementing the quaternion controller given by equation (12) and the position controller given by equation (14).

The figure 10 shows that X4-MaG was able to achieve very accurate positioning performances when hovering. The maximum error on the X and Y axes was only 3cm, with a standard deviation of only 0.77cm.

The figure 11a shows the step responses of the robot during displacements X, Y and Z in five different tests. The standard deviation of the tracking error was only 1.67cm in the case of X, 1.13cm in that of Y and 0.12cm in that of Z. The trajectories obtained were highly repeatable and accurate. A more complex trajectory was performed, as shown in figure 11b. In this case, the robot followed three 1-m circles tilted by 45° from the horizontal at a speed of $0.8 m.s^{-1}$. The robot followed the appropriate trajectory faithfully, since the standard deviation along the overall trajectory was only 2.9cm.

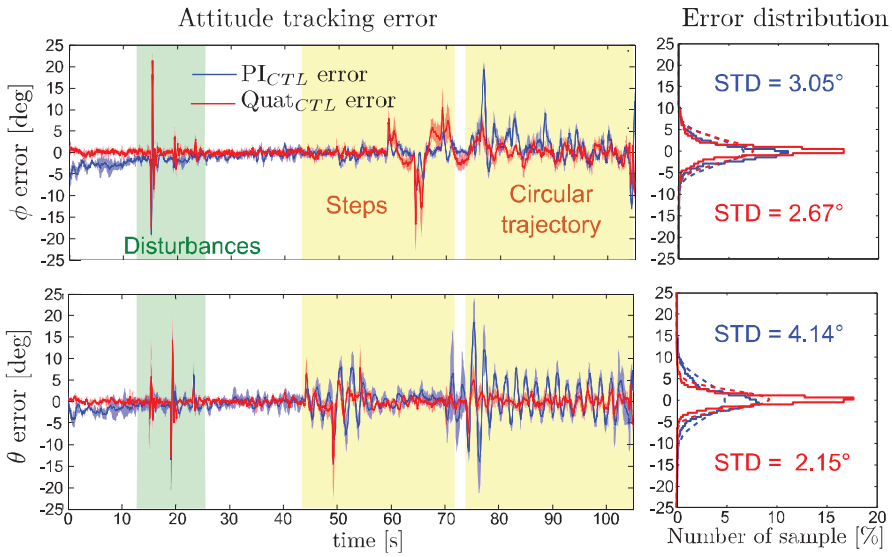

Figure 9: On the left, the mean attitude tracking error during a complex position trajectory (take-off, attitude disturbances, hovering, lateral steps, tilted circle and landing) recorded with each of the two controllers. The trajectory was done four times with each controller. The red and blue lines correspond to the mean attitude error made by the quaternion controller and the cascaded PI controller, respectively. The shaded areas correspond to the standard deviation of the various trajectories. On the right, the distribution of the tracking errors (continuous lines) and the corresponding Gaussian curves (dotted lines) recorded with each controller (blue for the cascaded PI controller, and red for the geometric quaternion-based controller.)

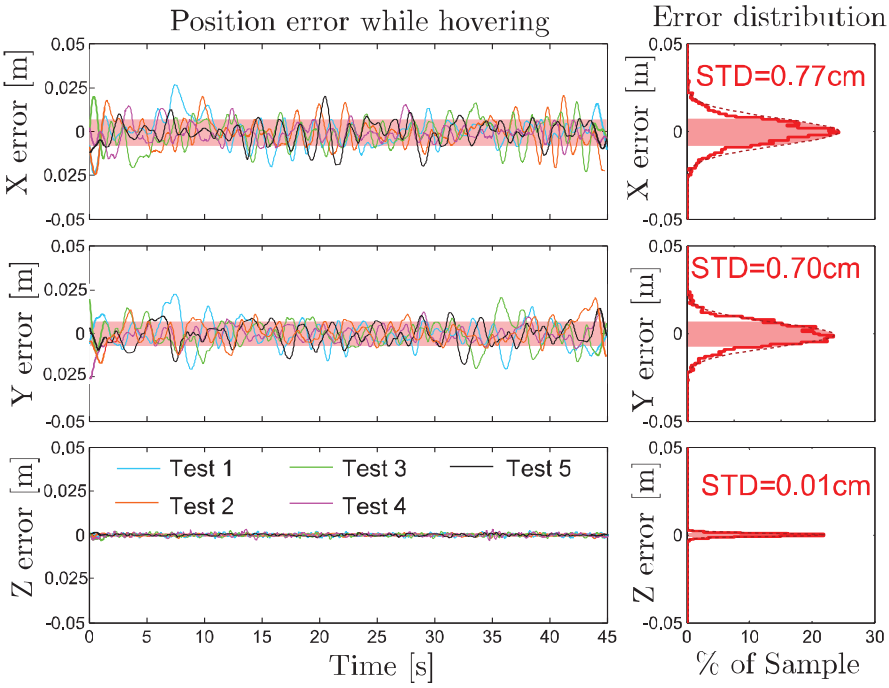

Figure 10: On the left, time course of positions X, Y and Z of X4-MaG while hovering in 5 different 45-second tests. Each test is plotted in a different color and the red shaded area corresponds to the mean standard deviation. The tracking performances obtained with the position controller described by equations (14-16) combined with the quaternion-based attitude controller led to a very accurate 3D positioning, since the standard deviation was only 0.77 cm and the maximum error recorded was less than 3cm.

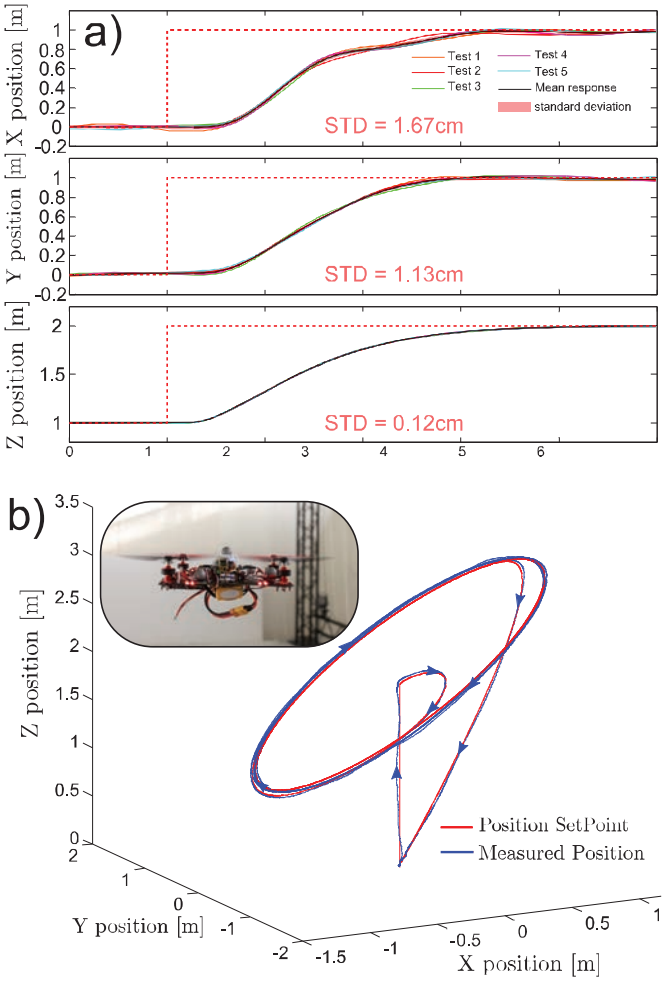

Figure 11: a) Step responses along all the three position axes in 5 tests. Each test is plotted in a different color: the black line gives the mean response and the blue shaded area corresponds to the standard deviation. b) Example of a circular 3-D trajectory tilted by 45° from the horizontal, achieved at a speed of $0.8 m.s^{-1}$. The standard deviation of the error recorded during the flight was as small as 2.9cm.

In addition, as shown in table 1, the robot was able to fly at $1.2 m.s^{-1}$ along an imposed 3-D circular path with a precision of 3.2cm (standard deviation), and the maximum error recorded was less than 8.3cm. These results are in the range of the results obtained in similar trajectories by ([20 and [22]) with the 500-gram Hummingbird quadrotor distributed by Ascending Technologies ([5]).

Table 1: Performances of the quadrotor for two different flying modes.

Flying Mode			Max Error	Mean Error	STD
Hover		x	2.8cm	0.03cm	0.7cm
		y	2.7cm	0.02cm	0.6cm
		z	0.4cm	0.00cm	0.3cm
Path Following	$0.8 m.s^{-1}$	x	5.3cm	0.04cm	1.7cm
		y	6.7cm	0.02cm	2.9cm
		z	3cm	0.14cm	1.5cm
Path Following	$1.2 m.s^{-1}$	x	8.3cm	0.3cm	2.3cm
		y	7.0cm	0.7cm	3.2cm
		z	6.6cm	0.5cm	2.0cm

5. CONCLUSION

The new open-source, open-hardware quadrotor called X4-MaG presented here can be assembled for a total cost of less than 500€. This quadrotor can be piloted either directly via an R/C radio emitter or from a ground station in an automatic mode. The Linux-based Computer-On-Module used as a high-level controller makes this robot highly versatile and customizable. A new formulation for the complementary filter and an attitude controller based on quaternions were developed, and their performances were compared with those of a classical cascaded PI attitude controller. It was established that the platform was able to perform both accurate hovering (STD $<$ 0.8cm) and faithful position tracking (STD $>$ 3.2cm) while effecting complex trajectories. The development of aerial autonomous robots is still at its infancy but it will certainly expand dramatically for at least the next decade. The development of open aerial robotic platforms will certainly facilitate the implementation of such platform for education and research purposes as well as industrial requirements. Documentation, tutorials, parts list and the whole software package are available at the X4-MaG project website: http://www.gipsa-lab.fr/projet/RT-MaG/.

Future works will consist of implementing onboard the free-flying X4-MaG, minimalistic bio-inspired strategies, such as those described in [23, 24, 25]. Such minimalist vision based strategies would be useful to make future insect-scale robots (e.g., [26]) relying on their own onboard sensors.

A Quadrotor physical parameters

Table 2: Quadrotor physical parameters.

Parameter	Description	Value	Unit
l	Half span of the robot	0.1	m
m	Mass of the robot	0.307	kg
I_{xx}	Inertia on X-axis	5.5×10^{-4}	$kg.m^2$
I_{yy}	Inertia on Y-axis	5.46×10^{-4}	$kg.m^2$
I_{zz}	Inertia on Z-axis	9.95×10^{-4}	$kg.m^2$
c_T	Thrust coefficient	1.121×10^{-6}	–
c_Q	Drag coefficient	8.895×10^{-8}	–

B Precision of the Vicon system in the flying arena

B.1 Introduction

The Vicon's localization system is one of the most used motion tracking system. However, it is difficult to find information about the accuracy and the precision of this system, because these two parameters strongly depend of the cameras' position and configuration. For example, [27] assessed that the accuracy of such a system was about 0.063*mm* in a most favorable configuration with a precision of about 0.015*mm*. [28] compared different localization systems and measured an accuracy of 0.24*mm*.

In order to analyze the results obtained with various airborne and ground robots in our flying arena, it was necessary to assess the precision of the Vicon's localization system.

First, it is proposed to define the difference between *precision* and *accuracy*. The detailed definition of these two terms available at [29] or [30] and can be summarized as follows:

The *accuracy* of a measurement system is the "closeness of agreement between a measured quantity value and a true quantity value of a measurand." (from [30])

The *precision* of a measurement system is "the closeness of agreement between indications or measured quantity values obtained by replicate measurements on the same or similar objects under specified conditions." (from [30])

According to these definitions, assessing the *accuracy* of the Vicon System may be quite a difficult task because it requires using another accurate measurement system. For example, even if one used a simple accurate measuring tape for this purpose, many precautions and complex procedures would be necessary because the measurements would have to be carried out in a large 3-D space; whereas the *precision* of the Vicon localization system can be assessed much more easily. It was therefore proposed in this study to determine the *precision* of the Vicon system used in our flying arena rather than the *accuracy* of this system.

B.2 Procedure and conditions

A given object was placed at various points in the flying arena for 30 seconds, and the data obtained with the Vicon system during this time were recorded. The structure holding the camera was motorized and could be moved up and down to adapt the "working volume" of the arena as required. All the results presented here were obtained with the maximum working volume ($6m \times 6m \times 8m$). The mean position measured, the standard deviation of the measurements, and the maximum deviation of each position in the arena were then calculated.

The object used in these tests was the Wand which is also a calibration tool provided by *Vicon Motion Systems Ltd* and endowed with IR reflectors. This object is shown in the figure 12 in various position during the procedure.

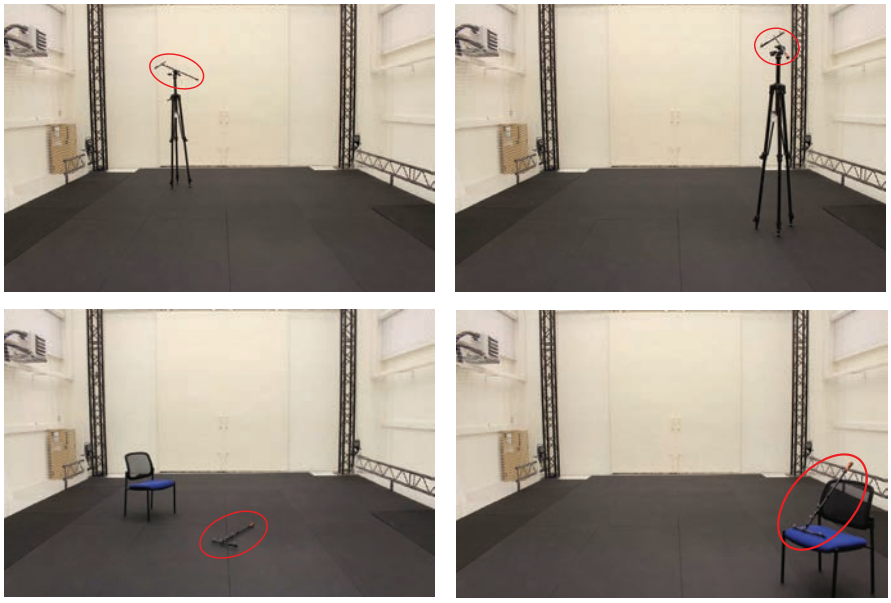

Figure 12: In order to assess the precision of the Vicon system, a given object (the "Wand" used to calibrate the system) was placed in various positions. Each of the Wand positions measured by the motion capture Vicon's system was recorded for 30 seconds. The Wand was placed in 9 different positions, each of which was repeated at 4 different heights, making 36 measurements in all, covering practically the whole volume of the flying arena. The Wand (surrounded by a red circle) is shown here at various points in the arena and at various heights.}

B.3 The metrics

The precision was assessed here using two different metrics, the **standard deviation** and the **maximum deviation**. The values of these parameters were determined at each of the Wand positions tested, and then globally. Complete volume and overall values were obtained using the following procedure: P series of measurements were performed for δt-seconds each with the Vicon system at a frequency f_s. Each position is denoted p_k, where $k \in \mathbb{N}$ and $1 \leq k \leq P$.

At each position p_k, we have N measurements denoted $x_{i,k}$ (where $N = \delta t.f_s$), giving a vector with the 3 coordinates $x_{i,k} = (X_{i,k} \quad Y_{i,k} \quad Z_{i,k})^T$. At each position p_k, the mean value $E(p_k)$, the standard deviation $\sigma(p_k)$ and the maximum deviation measured $\epsilon_{max}(p_k)$, are defined as follows:

$$E(p_k) = \frac{1}{N}\sum_{i=1}^{N} x_{i,k} \tag{17}$$

$$\sigma(p_k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_{i,k} - E(p_k)\right)^2} \tag{18}$$

$$\epsilon_{max}(p_k) = max\left(|x_{1.k} - E(p_k)|, |x_{2,k} - E(p_k)|, ..., |x_{N,k} - E(p_k)|\right) \quad (19)$$

By extension, the overall standard deviation, denoted $E(\sigma)$, can be defined as the mean standard deviation of all the positions. The overall maximum deviation, denoted ϵ_{MAX}, can also be defined, corresponding to the largest value of all the maximum deviations:

$$E(\sigma) = \frac{1}{P}\sum_{k=1}^{P}\sigma(p_k) \quad (20)$$

$$\epsilon_{MAX} = max\left(\epsilon_{max}(p_1), \epsilon_{max}(p_2), ..., \epsilon_{max}(p_P)\right) \quad (21)$$

B.4 Results

The above procedure was applied to 36 different positions (9 positions at 4 different heights, covering a large part of the whole volume of the flying arena). The data obtained at each position were recorded for 30 seconds at a frequency of 500Hz. The arena was equipped with 17 T-40s cameras arranged as shown in the figure 13. Before the experiment started, the calibration was repeated until an "image error" of less than 0.35 pixels was reached with each camera (in the Tracker 2.0 program), i.e., until the position of a marker's centroid could be measured with a precision greater than 0.35 pixels in the case of each camera (see figure 13).

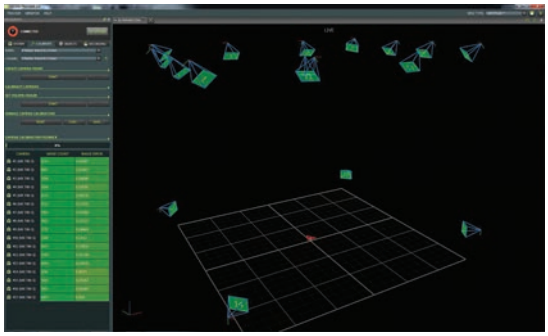

Figure 13: Screenshot of the Tracker software just before the experiment. The precision was assessed by the Tracker system in terms of the "image error", which is the estimated camera's positioning error in pixels. The use of multiple cameras makes it possible to achieve a sub-pixel resolution. Before all the experiments conducted in the arena, we checked that the calibration yielded an "image error" of less than 0.35 pixels.

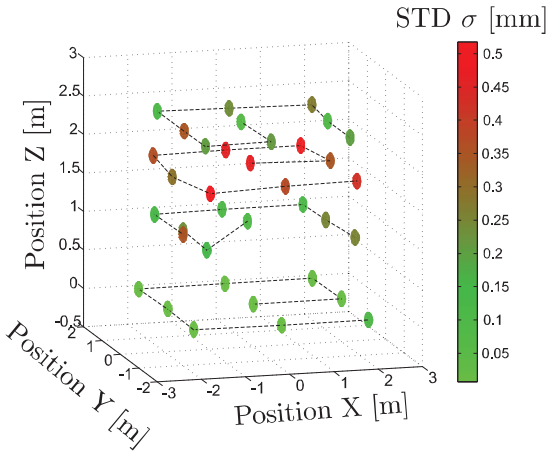

Figure 14: Diagram of the precision depending on the position of the object. The standard deviation given here is that of the "worst" of the X, Y or Z axes.

The figure 15a gives the Wand's position measured versus time in the 36 positions. The time series of each position is delimited using two 'x' markers. The figure 15b) is a zoom showing one of the time series, with the corresponding standard deviation (blue shaded area), the mean position (black line) and the maximum deviation (red circle).

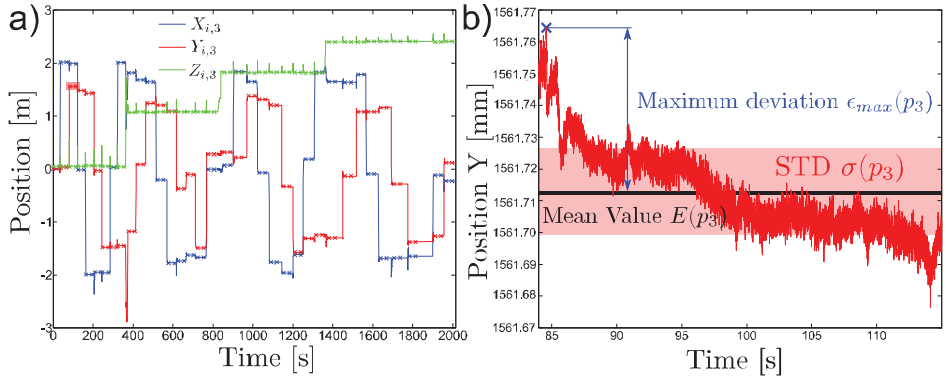

Figure 15: a) A recording of the 36 different positions delimited by 'x' markers on which the metrics were computed. The positions on the axes X, Y and Z are plotted in blue, red and green respectively. Peaks between two different positions in the recordings were due to the manual displacement of the Wand. b) Zoom corresponding to the measurements recorded at the position p_3 (corresponding to the red shaded area in figure a). The red line gives the position measured by the Vicon system with time, and the red shaded area corresponds to the standard deviation $\sigma(p_3)$. The blue crosses correspond to the maximum deviation $\epsilon_{max}(p_3)$ measured with respect to the mean position $E(p_3)$, which is plotted in the form of a black line.

The table 3 gives the standard deviations and maximum deviations measured at 6 different positions (chosen at random among the 36 positions tested). The values of these two parameters in the other positions were also computed, but have not been presented here for the sake of clarity. The precision varied because the number of cameras able to detect the object varied, depending on the object's position p_k. This situation is also illustrated in the figure 14, where colored spheres indicate the precision at a given position. The lowest precision was observed at a height of about 1.8 m because the four cameras located at the bottom of the arena could not detect the object, which was also relatively far from the camera located at the top of the arena.

Table 3: The standard deviation and maximum deviation measured at 6 of the 36 positions tested. a) The standard deviation (in millimetres) on each of the X, Y and Z axes. b) The maximum deviation measured with respect to the mean value (in millimetres) on each of the X, Y and Z axes.

a)	Standard deviation $\sigma(p_k)$ $[mm]$					
	p_{34}	p_{17}	p_{29}	p_5	p_{15}	p_{33}
X	0.1098	0.0970	0.0569	0.0148	0.1916	0.3442
Y	0.2063	0.1457	0.0816	0.0105	0.3536	0.0251
Z	0.1037	0.0580	0.0708	0.0052	0.0992	0.0988
b)	Maximum measured deviation $\epsilon_{max}(p_k)$ $[mm]$					
	p_{34}	p_{17}	p_{29}	p_5	p_{15}	p_{33}
X	0.2791	0.2554	0.1267	0.0755	0.5252	0.8023
Y	0.5453	0.3382	0.1727	0.0548	0.8859	0.0452
Z	0.2395	0.1313	0.1759	0.0231	0.3424	0.2236

The table 4 gives the minimum standard deviation (the greatest precision), the maximum standard deviation (the lowest precision) and the mean standard deviation (the average precision with the volume tested) obtained on each axis using the data recorded on the 36 different positions. These values reflect the global standard deviation, the global maximum standard deviation and global maximum deviation . It can be seen from this table that the maximum standard deviation was about 0.46*mm* on the X-axis, 0.52*mm* on the Y axis and 0.19*mm* on the Z-axis. The mean standard deviation measured was about 0.14*mm* on the X-axis, 0.15*mm* on the Y-axis and 0.07*mm* on the Z-axis. In both cases, it can be reasonably assumed that the precision was greater than 1mm, i.e., that the variations in the measurements (the precision) were in the sub-millimetric range.

Table 4: The minimum standard deviation observed in the arena was found to be about 0.0025*mm*, which was probably greatly underestimated; this value cannot be valid because it is well below the *accuracy* ("trueness") of the Vicon system. The maximum standard deviation observed in the arena was about 0.52*mm*, which was highly accurate. The mean standard deviation observed can be said to be less than 0.2*mm*. The maximum deviation measured was less than 1.5*mm* (1.31*mm*), which means that the position of the object was given by the Vicon system with a precision of $\pm$ of 1.5*mm* in the worst case.

	Min STD $\min(\sigma(p_k))$ $[mm]$	Max STD $\max(\sigma(p_k))$ $[mm]$	Mean STD $E(\sigma)$ $[mm]$	Max deviation ϵ_{MAX} $[mm]$
X	0.0025	0.46209	0.1375	1.1187
Y	0.0033	0.5173	0.1492	1.3057
Z	0.0042	0.1889	0.0714	0.3653

B.5 Discussion

Based on the results presented above, it can be concluded that the greatest positioning error observed here (in terms of maximum deviation measured with respect to the mean value) amounted to less than 1.5*mm*. i.e., the lowest precision of the measurements obtained was $\pm$1.5*mm*. A parameter more commonly used to assess the precision is the standard deviation. The mean precision of the Vicon system observed in the flying arena in its largest configuration was as small as 0.$*mm* in terms of the mean standard deviation and 0.6mm in terms of the greatest standard deviation (the worst case). In both cases, it can be assumed that the precision of the system tested was in the sub-millimetric range.

These results obviously depended on the positions of the cameras, the quality of the calibration and several other parameters. For example, all the cameras were supported by a motorized metallic structure which made it possible to adapt the "working" space. Since the metallic structure was relatively large, its position necessarily depended on the current temperature of the arena. To deal with this problem, the temperature in the arena was regulated during the experiments, and calibrations were performed at least twice a day (or more frequently if the precision was affected). In addition, the required level of precision could presumably be achieved only if some experimental conditions were met and the calibration was performed properly by repeating it until a maximum error of less than 0.35 pixels was reached in the *Tracker* program (see figure 13. In short, a satisfactory level of precision can be achieved with this set-up only if the following conditions are met:

- The temperature must not change during the experiment ($\pm$ 1°),
- The calibration must have a precision of at least 0.35 pixels in the worst case (in that of the least well calibrated camera),
- The ambient light must be kept constant, especially in the infra-red band: the awnings/roller blinds of the arena were kept down during the present experiments to prevent the sunlight from disturbing the measurements.

As the levels of precision given in this paper were obtained with the largest configuration of the arena, it can be reasonably assumed that better (or at least equivalent) performances would be obtained with smaller configurations. In conclusion, the precision of our Vicon system, composed by 17 cameras (T40S), as a means of localizing objects in our flying arena can be said to be in the sub-millimetre range (0.6*mm* in the case of the largest standard deviation).

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their fruitful comments, M. Boyron and J. Diperi for their great work on the quadrotor, F. Expert for his help with the experiments, and F. Colonnier, J. Dumon and J. Minet for their fruitful discussions. This work was supported by CNRS, Aix-Marseille University, the Provence-Alpes-Côte d'Azur region and the French National Research Agency (ANR) with the EVA, IRIS projects (EVA and IRIS project under ANR grant numbers ANR608-CORD-007-04 and ANR-12-INSE- 0009, respectively) and the Equipex/Robotex project.

REFERENCES

[1] P. C. Garcia, R. Lozano, A. E. Dzul, Modelling and Control of Mini-Flying Machines, Springer, 2006.

[2] K. P. Valavanis, Advances in Unmanned Aerial Vehicles: *State of the Art and the Road to Autonomy*, Vol. 33, Springer, 2008.

[3] R. Mahony, V. Kumar, P. Corke, Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor, *IEEE Robot. Automat. Mag.* 19 (3) (2012) 20–32. doi:10.1109/MRA.2012.2206474.

[4] M.-D. Hua, T. Hamel, P. Morin, C. Samson, Introduction to Feedback Control of Under- actuated VTOL Vehicles: A Review of Basic Control Design Ideas and Principles, Control Systems, *IEEE* 33 (1) (2013) 61–75. doi:10.1109/MCS.2012.2225931.

[5] Ascending Technologies GmbH, Hummingbird and Pelican multicopters (2011).
URL http://www.asctec.de/uav-applications/research/products/

[6] A. Kushleyev, D. Mellinger, C. Powers, V. Kumar, Towards a Swarm of Agile Micro Quadrotors, *Autonomous Robots* 35 (4) (2013) 287–300. doi:10.1007/s10514-013-9349-9.
URL http://dx.doi.org/10.1007/s10514-013-9349-9

[7] R. Konomura, K. Hori, Designing Hardware and Software Systems Toward Very Compact and Fully Autonomous Quadrotors, in: Advanced Intelligent Mechatronics (AIM), 2013 *IEEE/ASME International Conference on*, 2013, pp. 1367–1372. doi:10.1109/AIM.2013.6584285.

[8] R. Spica, P. Robuffo Giordano, M. Ryll, H. H. Bülthoff, A. Franchi, An Open-Source Hardware/Software Architecture for Quadrotor UAVs, in: 2nd Workshop on Research, Education and Development of Unmanned Aerial System, Compiègne, France, 2013.
URL http://hal.inria.fr/hal-00906138

[9] C. Lehnert, P. Corke, μAV - Design and Implementation of an Open Source Micro Quadrotor, 2013.

[10] OpenPilot, Open autopilot for multi-rotor craft, helicopters, and fixed wing aircraft (Jan. 2010).
URL http://www.openpilot.org/

[11] MultiWii, General purpose software to control a multirotor rc model (Dec. 2010).
URL http://www.multiwii.com/

[12] H. Lim, J. Park, D. Lee, H. J. Kim, Build Your Own Quadrotor: Open-Source Projects on Unmanned Aerial Vehicles, *IEEE Robot. Automat. Mag.* 19 (3) (2012) 33–45. doi:10.1109/MRA.2012.2205629.

[13] A. Manecy, RT-MaG Project (Jan. 2014). URL http://www.gipsa-lab.fr/projet/RT-MaG/

[14] R. Mahony, T. Hamel, J.-M. Pflimlin, Nonlinear Complementary Filters on the Special Orthogonal Group, Automatic Control, *IEEE Transactions on* 53 (5) (2008) 1203 –1218.
doi:10.1109/TAC.2008.923738.

[15] T. Lee, M. Leoky, N. McClamroch, Geometric Tracking Control of a Quadrotor UAV on SE(3), in: Decision and Control (CDC), 2010 49th *IEEE Conference on*, 2010, pp. 5420–5425. doi:10.1109/CDC.2010.5717652.

[16] D. Cabecinhas, R. Cunha, C. Silvestre, A nonlinear quadrotor trajectory tracking controller with disturbance rejection, *Control Engineering Practice* 26 (0) (2014) 1 – 10. doi:http://dx.doi.org/10.1016/j.conengprac.2013.12.017.
URL http://www.sciencedirect.com/science/article/pii/S0967066114000343

[17] Vicon's motion capture system. URL http://www.vicon.com/Software/Tracker

[18] P. Bake, Flyduino multicoptershop (2011). URL http://flyduino.net/

[19] D. Mellinger, V. Kumar, Minimum Snap Trajectory Generation and Control for Quadrotors, in: Robotics and Automation (ICRA), 2011 *IEEE International Conference on*, 2011, pp. 2520–2525. doi:10.1109/ICRA.2011.5980409.

[20] N. Michael, D. Mellinger, Q. Lindsey, V. Kumar, The GRASP Multiple Micro-UAV Testbed, Robotics Automation Magazine, *IEEE* 17 (3) (2010) 56–65. doi:10.1109/MRA. 2010.937855.

[21] Quanser, QUARC? 2.0, Soft Real-Time Control Software for Windows (2009).
URL http://www.quanser.com/Products/quarc

[22] D. Mellinger, N. Michael, V. Kumar, Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors, *The International Journal of Robotics Research* 31 (5) (2012) 664–674.

[23] F. Expert, F. Ruffier, Controlling Docking, Altitude and Speed in a Circular High-Roofed Tunnel thanks to the Optic Flow, in: Intelligent Robots and Systems (IROS), 2012 *IEEE/RSJ International Conference on*, 2012, pp. 1125–1132. doi:10.1109/IROS.2012. 6385946.

[24] A. Manecy, N. Marchand, S. Viollet, Hovering by Gazing: a Novel Strategy for Implementing Saccadic Flight-Based Navigation in GPS-Denied Environments, *International Journal of Advanced Robotic Systems* 11 (66) (2014) . doi:10.5772/58429.

[25] F. L. Roubieu, J. R. Serres, F. Colonnier, N. Franceschini, S. Viollet, F. Ruffier, A biomimetic vision-based hovercraft accounts for bees' complex behaviour in various corridors, *Bioinspiration & biomimetics* 9 (3) (2014) 036003. doi:10.1088/1748-3182/9/3/ 036003.

[26] K. Y. Ma, P. Chirarattananon, S. B. Fuller, R. J. Wood, Controlled Flight of a Biologically Inspired, Insect-Scale Robot, *Science* 340 (6132) (2013) 603–607. arXiv:http://www.sciencemag.org/content/340/6132/603.full.pdf, doi:10.1126/science.1231806. URL http://www.sciencemag.org/content/340/6132/603.abstract

[27] M. Windolf, N. G¨otzen, M. Morlock, Systematic accuracy and precision analysis of video motion capturing systems-exemplified on the Vicon-460 system, *Journal of Biomechanics* 41 (12) (2008) 2776 – 2780. doi:http://dx.doi.org/10.1016/j.jbiomech.2008.06.024. URL http://www.sciencedirect.com/science/article/pii/S0021929008003229

[28] Nippon Engineering College in Tokyo, Results:basic measurement accuracy, and processing time (July 2002). URL http://www.ne.jp/asahi/gait/analysis/comparison2002/Result/basic/basic_eng.html

[29] F. E. Grubbs,Errors of Measurement,Precision, Accuracy and the Statistical Comparison of Measuring Instruments, *Technometrics* 15 (1) (1973) 53–66. arXiv:http://www.tandfonline.com/doi/pdf/10.1080/00401706.1973.10489010, doi:10.1080/00401706.1973.10489010. URL http://www.tandfonline.com/doi/abs/10.1080/00401706.1973.10489010

[30] International Vocabulary of Metrology - Basic and General Concepts and Associated Terms, VIM 3rd Edition, Vol. JCGM 200:2012.
URL http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2012.pdf