Quadrotor Quaternion Control

Conference Paper · June 2015

DOI: 10.1109/ICUAS.2015.7152367

CITATIONS
69

READS
12,876

3 authors:

Jossué Cariño Escobar
Center for Research and Advanced Studies of the National Polytechnic Institute
9 PUBLICATIONS **111** CITATIONS
SEE PROFILE

Hernán Abaunza González
Tecnológico de Monterrey
31 PUBLICATIONS **367** CITATIONS
SEE PROFILE

Pedro Castillo Garcia
University of Technology of Compiègne
195 PUBLICATIONS **4,348** CITATIONS
SEE PROFILE

Quadrotor Quaternion Control

J. Cariño*, H. Abaunza, P. Castillo

***Abstract*— This paper presents the design and practical implementation of a quaternion control scheme to globally stabilize a quadrotor aerial vehicle. First an attitude control law is proposed to stabilize the vehicle's heading, then a position control law is proposed to stabilize the vehicle in all its states. Using the position references, a smooth trajectory is calculated for the attitude controller to follow so that the position of the vehicle becomes stable. The proposed control law is such that the quadrotor system can be analyzed and controlled as a linear system. This model and control law are then numerically simulated to corroborate the closed-loop system's stability. At last, experimental flight tests were performed to validate practical results.**

I. Introduction

A quadrotor multi-copter is a popular platform in UAV (Unmanned Aerial Vehicles) research because of the relatively simple dynamic model and its now low-cost implementation. Although the dynamic model could be considered simple, the non-linearities introduced by using an Euler's angles representation makes the analysis of the system and the design of control laws cumbersome. The Euler's angles approach is preferred because of its intuitiveness, in [1], the authors describe the design, modeling, control, and simulation of a quadrotor by using this approach, but it has become apparent that it makes the platform's model loose its mathematical simplicity. A more stable and more simple mathematical tool for modeling these dynamics are quaternions, in [2], the authors highlight that "a quaternion-based controller can exhibit an unwinding phenomenon, where the controller unnecessarily rotates the attitude through large angles".

Quaternions are a type of hyper-complex numbers that can be used to describe rotations in 3D space. Its computational stability comes from the fact that only unit quaternions are required to describe rotations, reducing numeric errors when they are normalized. They lack the gimball-lock effect inherent to Euler's angles because of its relationship to the axis-angle representation. Their mathematical simpleness comes from the fact that no trigonometric functions are required directly when they are used to model a rigid body's dynamics, only arithmetic functions like quaternion products (which are computationally efficient).

In [3] the authors build a quaternion-based model for an hexa-rotor vehicle, which can be extended to build a similar model for a quad-rotor UAV. In [4], the authors build a quaternion-based model and control for a robot manipulator. In [5], [6], [7], and [4], the authors establish the mathematical tools for quaternion algebra and quaternion kinematics, in [8], the authors proposed a linear control using dual quaternions, however their example can only be applied to fully actuated systems, which is not the case in multi-copter vehicles.

The main contribution of this article is the separation of the quadcopter system into two analogous linear subsystems that can be stabilized using only state feedbacks. The close relationship of quaternions with the axis-angle representation by means of the Euler-Rodrigues rotation formula allows the proposal of an equivalent model between both representations. The nonlinearities on the axis-angle representation can be canceled using feedback linearization in order to have a linear attitude model. As the model using the axis-angle representation is analogous to the quaternion system, any control law and stability proof implemented on the first system has an equivalent response in the quaternion model. A state feedback is analyzed on the linear attitude model and then implemented on the platform using the quaternion representation. Once an attitude control law is implemented, a trajectory relying on the position and linear velocity can be obtained that guarantees global stability and then fed to the attitude control, resulting in a control law that fully stabilizes the system.

The paper is organized as follows: a Quaternions background and the dynamical model using this approach are given in section II, the control algorithms is introduced in section III, the main graphs representing the behavior of the closed-loop system are depicted in section IV, a quadrotor prototype and main sensors are described in section V, an experimental validation in a real platform is presented in section VI and finally some discussions about this work is given in section VII.

II. Quaternion Modeling

A. Background

Quaternions, which belong to the quaternion space $\mathbb{H}$, can be represented in many ways, but the one presented here is as a sum of a scalar component along an imaginary vector (see [5], [4], [7]).

$$\begin{aligned} &\boldsymbol{q} \in \mathbb{H}; \overline{q} \in \mathbb{R}^3; q_0 \in \mathbb{R} \\ &\boldsymbol{q} = q_0 + \overline{q} = q_0 + \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \end{aligned} \tag{1}$$

J. Cariño and H. Abaunza are with the UMI LAFMIA CNRS 3175, CINVESTAV-IPN, México, D. F. E-mail: (jcarino, habaunza)@cinvestav.mx

P. Castillo is with the Heudiasyc Lab. 7253 CNRS, Université de Technologie de Compiegne, France. E-mail: castillo@hds.utc.fr.

where $\boldsymbol{q}$ is a given quaternion, $\bar{q}$ is the complex vectorial part of $\boldsymbol{q}$, and q_0 is the scalar part of $\boldsymbol{q}$.

The main operation in quaternion algebra is the quaternion product, which is defined as

$$\begin{array}{l} \boldsymbol{r} \in \mathbb{H};\, \overline{r} \in \mathbb{R}^3;\, r_0 \in \mathbb{R} \\ \boldsymbol{q} \otimes \boldsymbol{r} = (q_0\, r_0 - \overline{q} \cdot \overline{r}) + (r_0\, \overline{q} + q_0\, \overline{r} + \overline{q} \times \overline{r}) \end{array} \tag{2}$$

The quaternion conjugate $\boldsymbol{q}^*$ is defined as $\boldsymbol{q}^* = q_0 - \overline{q}$. The quaternion norm, $||\boldsymbol{q}|| \in \mathbb{R}$, is defined as

$$||\boldsymbol{q}|| = \sqrt{\boldsymbol{q} \otimes \boldsymbol{q}^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{3}$$

The quaternion inverse is obtained from $\boldsymbol{q}^{-1} = \dfrac{\boldsymbol{q}^*}{||\boldsymbol{q}||}$. In this work, only unit quaternions are used for the attitude representation, thus, $\boldsymbol{q}^{-1} = \boldsymbol{q}^*$. Similarly, the quaternion natural logarithm mapping is defined as

$$\ln \boldsymbol{q} = \begin{cases} \ln ||\boldsymbol{q}|| + \dfrac{\overline{q}}{||\overline{q}||} \arccos \dfrac{q_0}{||\boldsymbol{q}||}, & ||\overline{q}|| \neq 0 \\ \ln ||\boldsymbol{q}||, & ||\overline{q}|| = 0 \end{cases} \tag{4}$$

For unit quaternions the above logarithmic mapping is reduced to

$$\ln \boldsymbol{q} = \begin{cases} \dfrac{\overline{q}}{||\overline{q}||} \arccos q_0, & ||\overline{q}|| \neq 0 \\ 0, & ||\overline{q}|| = 0 \end{cases} \tag{5}$$

This mapping is useful to change any unit quaternion to the axis angle representation, in which the quaternion is represented as an axis over which the vehicle will rotate, and a rotation angle.

$$\frac{\overline{\theta}}{2} = \ln \boldsymbol{q}, \quad \dot{\overline{\theta}} = \omega$$

where $\bar{\theta} \in \mathbb{R}^3$ is the vector which contains the axis-angle representation, in which $\frac{\bar{\theta}}{||\bar{\theta}||}$ describes the unitary axis over which the rotation is applied, $||\bar{\theta}||$ represents the magnitude of the rotation, and ω is the vehicle's angular velocity.

1) Quaternion Rotation: The rotation of a 3D vector from one reference frame to another can be accomplished with the three quaternion product

$$\begin{array}{l} ||\boldsymbol{q}|| = 1;\, v, v' \in \mathbb{R}^3 \\ v' = L\,(\boldsymbol{q}, v) = \boldsymbol{q}^* \otimes v \otimes \boldsymbol{q} \end{array} \tag{6}$$

The operator $L\,(\boldsymbol{q}, v)$ is linear as it does not change the magnitude of v.

According to Euler's theorem for rigid bodies [9], the relationship between the rotation of a body represented by a 3D axis and a rotation amount around that axis, and the quaternion representation of this rotation is:

$$\boldsymbol{q} = \cos\left(\frac{\theta}{2}\right) + \overline{e}\, \sin\left(\frac{\theta}{2}\right) \tag{7}$$

where $\overline{e} \in \mathbb{R}^3$ represents a unitary 3D axis in which the rigid body is rotated and $\theta \in \mathbb{R}$ is the amount of rotation. It can be seen that $||\boldsymbol{q}|| = 1$, thus this quaternion can be used in the rotation operator.

2) Quaternion kinematics: The kinematics of a rigid body in terms of its orientation and the angular velocity are given in [9] by

$$\dot{\boldsymbol{q}} = \frac{1}{2}\, \boldsymbol{q} \otimes \omega \tag{8}$$

B. Quadrotor Quaternion Dynamic Model

The dynamic model of a quadcopter using Newton-Euler equations with quaternions is described as follows

$$\dot{x} = \frac{d}{dt} \begin{bmatrix} p \\ \dot{p} \\ \boldsymbol{q} \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{p} \\ \boldsymbol{q} \otimes \dfrac{F_{th}}{m} \otimes \boldsymbol{q}^* + \overline{g} \\ \dfrac{1}{2}\, \boldsymbol{q} \otimes \omega \\ J^{-1}\, (\tau - \omega \times J\, \omega) \end{bmatrix} \tag{9}$$

where $p \in \mathbb{R}^3$ and $\dot{p} \in \mathbb{R}^3$ are the position and velocity vectors with respect to the inertial frame, F_{th} defines the thrust vector generated by the quadcopter motors, m and $\overline{g}$ represent the vehicle's mass and gravity vector, respectively, $\boldsymbol{q}$ describes the quaternion that represents the vehicle orientation with respect to the inertial frame, ω denotes the angular velocity of the quadcopter with respect to the body-fixed frame, J introduces the inertia matrix with respect to the body-fixed frame and $\tau = \tau_u + \tau_{ext}$, where τ_u and τ_{ext} are the input and external torques respectively, applied on the aerial vehicle in the body-fixed frame [3].

The relationships between the input torques and forces is

$$\begin{bmatrix} F_{th} \\ \tau_{u_x} \\ \tau_{u_y} \\ \tau_{u_z} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{4} k_i\, \omega_i^2 \\ l\left(k_1\, \omega_1^2 - k_2\, \omega_2^2 - k_3\, \omega_3^2 + k_4\, \omega_4^2\right) \\ l\left(k_1\, \omega_1^2 + k_2\, \omega_2^2 - k_3\, \omega_3^2 - k_4\, \omega_4^2\right) \\ \sum_{i=1}^{4} \tau_i\, (-1)^i \end{bmatrix}$$

where $k_i\, \omega_i^2$ defines the thrust of the propeller of motor i with respect to its angular velocity ω_i, l is the distance from the center of mass to the motor axis of action and τ_i denotes the torque of motor i.

III. Control algorithms

A. Attitude algorithm

From (9) and using (5), the attitude dynamic of the aerial vehicle could be expressed as

$$\frac{d}{dt} \begin{bmatrix} \overline{\theta} \\ \omega \end{bmatrix} = \begin{bmatrix} \omega \\ J^{-1}\, (\tau - \omega \times J\, \omega) \end{bmatrix} \tag{10}$$

where $\overline{\theta} = 2\, \ln \boldsymbol{q}$. Observe that the terms J^{-1}, and $(\omega \times J\, \omega)$ can be compensated using an appropriate $\tau = J\, u + \omega \times J\, \omega$ with u as the new control input. Thus, the previous model becomes:

$$\begin{aligned} \frac{d}{dt} \begin{bmatrix} \overline{\theta} \\ \omega \end{bmatrix} &= \begin{bmatrix} \omega \\ u \end{bmatrix} = \begin{bmatrix} \overline{0} & I_3 \\ \overline{0} & \overline{0} \end{bmatrix} \begin{bmatrix} \overline{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} \overline{0} \\ I_3 \end{bmatrix} u \\ &= A\, x_{att} + B\, u \end{aligned} \tag{11}$$

where x_{att} is the attitude of the vehicle. Proposing

$$u = -K_{att}\,(x_{att} - x_{att_d}) \tag{12}$$

where $K_{att} > 0$ denotes the gains and x_{att_d} the desired attitude. Observe that this control law makes system (11) exponentially stable.

B. Position strategy

From (9), it follows that

$$\dot{x}_{pos} = \begin{bmatrix} \dot{p} \\ \boldsymbol{q} \otimes \dfrac{F_{th}}{m} \otimes \boldsymbol{q}^* + \overline{g} \end{bmatrix} \tag{13}$$

where $x_{pos} = \begin{bmatrix} p^T & \dot{p}^T \end{bmatrix}^T$. Notice that the quaternion norm is $||\boldsymbol{q}|| = 1$ thus, the direction of the input vector $u_p = \boldsymbol{q} \otimes \dfrac{F_{th}}{m} \otimes \boldsymbol{q}^*$ depends exclusively on the vehicle's orientation, and remark that its magnitude depends entirely of the magnitude of the vector $\frac{F_{th}}{m}$, which is also a control input of the quadcopter. The algebraic relationship between $F_{th}, \boldsymbol{q}$ and u_p is:

$$\begin{aligned} \boldsymbol{q}' &= (b \cdot u_p + ||u_p||) + b \times u_p \\ \boldsymbol{q} &= \frac{\boldsymbol{q}'}{||\boldsymbol{q}'||} \\ F_{th} &= b\,||u_p||\,m \end{aligned} \tag{14}$$

where b is a unitary vector denoting the axis in which the thrust acts in the body frame.

Let u_{p_d} be a desired trajectory that stabilizes system (13) if $u_p \cong u_{p_d}$. Substituting u_p for u_{p_d} in equations (14) gives us the quaternion trajectory $\boldsymbol{q}_d$ and thrust vector trajectory F_{th_d} that the quadcopter must follow in order for $u_p \cong u_{p_d}$. As the thrust vector is a control input, it is defined as $F_{th} \coloneqq F_{th_d}$.

The angular velocity trajectory can be obtained from $\omega_d = \dfrac{d}{dt}(2 \ln \boldsymbol{q}_d)$. Therefore, the desired attitude becomes $x_{att_d} = \begin{bmatrix} (2 \ln \boldsymbol{q}_d)^T & \omega_d^T \end{bmatrix}^T$.

From the attitude control observe that K_{att} can be chosen such that $\boldsymbol{q} \cong \boldsymbol{q}_d$, which implies that $u_p \cong u_{p_d}$ and then, the model (13) can be rewritten as

$$\dot{x}_{pos} = \begin{bmatrix} \dot{p} \\ u_{p_d} + \overline{g} \end{bmatrix} \tag{15}$$

Choosing u_{p_d} such that it compensates the gravity vector $u_{p_d} = u_{pos} - \overline{g}$, this results in another linear representation for the position subsystem

$$\begin{aligned} \dot{x}_{pos} &= \begin{bmatrix} \overline{0} & I_3 \\ \overline{0} & \overline{0} \end{bmatrix} x_{pos} + \begin{bmatrix} \overline{0} \\ I_3 \end{bmatrix} u_{pos} \\ &= A_{pos}\, x_{pos} + B_{pos}\, u_{pos} \end{aligned} \tag{16}$$

with u_{pos} denoting the new input. Proposing

$$u_{pos} = -K_{pos}\,(x_{pos} - x_{pos_d}) \tag{17}$$

with x_{pos_d} as the desired position state, then this controller exponentially stabilizes subsystem (16).

IV. Simulations results

The closed-loop system was validated in simulation using the Python programming language, along with the SciPy [10], NumPy [11] and Python Systems Control [12] libraries. The quadrotor model (9) was simulated by integrating numerically $\dot{x}$ using SciPy's *integrate* function for a period of ten seconds. The control inputs (12) and (17) were implemented using K_{att} and K_{pos}, which were determined using a linear quadratic regulator using the function *lqr()* included in the python control library:

$$K_{att} = \begin{bmatrix} 100 & 0 & 0 & 14.142 & 0 & 0 \\ 0 & 100 & 0 & 0 & 14.142 & 0 \\ 0 & 0 & 3.16\mathrm{e}-6 & 0 & 0 & 2.51e-3 \end{bmatrix}$$

$$K_{pos} = \begin{bmatrix} 3.1622 & 0 & 0 & 2.7063 & 0 & 0 \\ 0 & 3.1622 & 0 & 0 & 2.7063 & 0 \\ 0 & 0 & 3.1622 & 0 & 0 & 2.7063 \end{bmatrix}$$

The quaternion operations were previously defined using NumPy's vector and cross products, and then included in a custom library. The parameters used were the following.

$$\begin{aligned} J &= \begin{bmatrix} 0.0118976 & 0 & 0 \\ 0 & 0.0118976 & 0 \\ 0 & 0 & 0.0218816 \end{bmatrix} \\ m &= 1.3\,kg \\ p_0 &= \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^T \\ \dot{p}_0 &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \\ \boldsymbol{q}_0 &= 1 + \overline{0} \\ \omega_0 &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \\ p_d &= \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^T \\ \dot{p}_d &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \end{aligned}$$

1) Attitude Stabilization: The first simulation were defined to stabilize the attitude of the quadrotor. Figures 1 and 2 show the stabilization of the vehicle states, for example in Figure 1, it can be seen that the quaternion stabilizes in the equilibrium point. Similarly, in Figure 2 we can observe

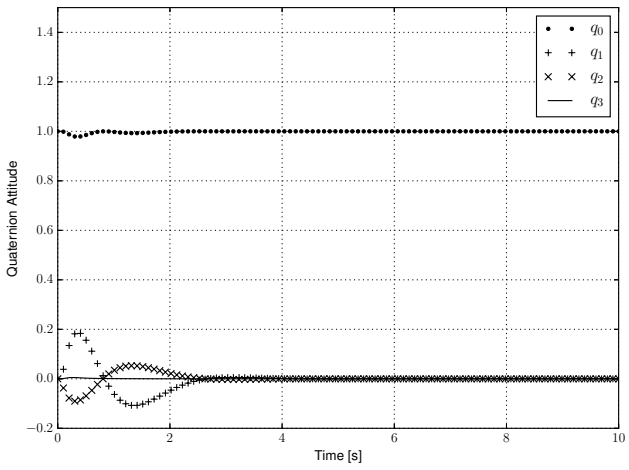

Fig. 1. Quadcopter attitude (q).

that the angular velocities converge to zero. This means that the vehicle will converge exponentially to a given orientation and stay still in it.

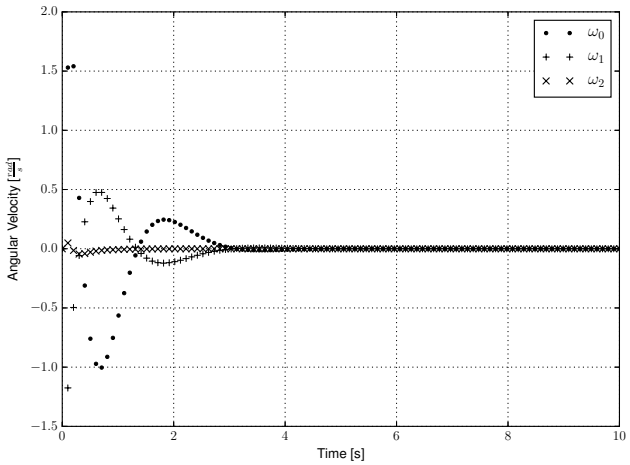

Fig. 2. Quadcopter rotational velocity ω.

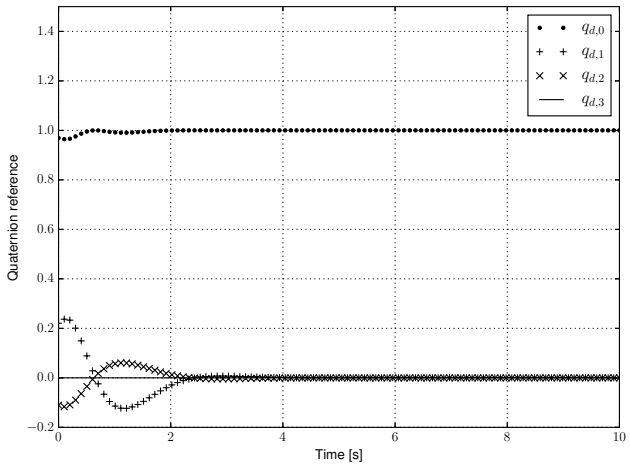

Fig. 3. Quadcopter attitude reference (q_d).

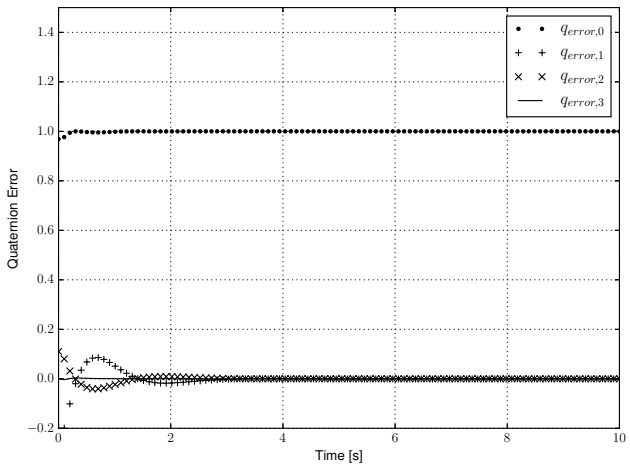

Fig. 4. Quadcopter attitude error (q_e).

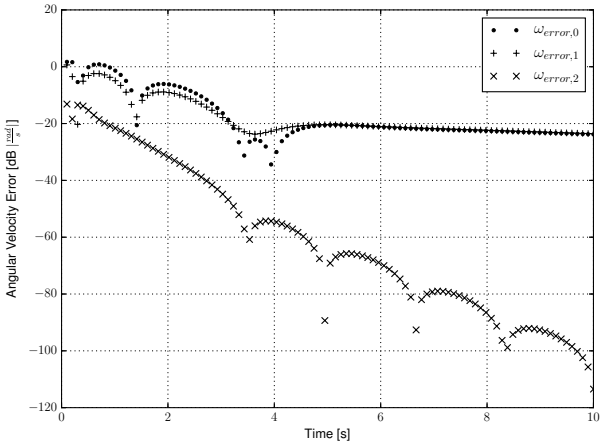

Fig. 5. Quadcopter attitude velocity error (ω_e).

A trajectory is calculated from the position control (see section IV-.2) $\boldsymbol{q}_d$ comes from the position control, the desired attitude is illustrated with $\boldsymbol{q}_d$ in Figure 3. Figure 4 illustrates the behavior of the quaternion error. Since the quaternion error is defined as: $\boldsymbol{q}_e = \boldsymbol{q} \otimes \boldsymbol{q}_d^{-1}$, so in this case the attitude $(\boldsymbol{q})$ converges to the desired trajectory$(\boldsymbol{q}_d)$, making the error $\boldsymbol{q}$ converge to the unit quaternion.

On the other hand, Figure 5 depicts the angular velocity errors in a logarithmic scale, it can be seen that the angular velocity converges to zero at an infinite time, since the error values approach to $-\infty$, and the inverse of the logarithmic scale $e^{-\infty} \to 0$. In practical terms it can be considered that the system's angular velocity really converges to zero.

2) Translational Stabilization: An initial position $p_0 = [1, 2, 0]$ was given, then a desired equilibrium point $p_d = [0, 0, 1]$ was proposed as a reference to globally stabilize the system.

Figures 6 and 7 show that the system's position exponentially converges from the initial condition p_0 to the reference p_d.

Figures 8 and 9 introduce the translational error in a logarithmic scale, we can see that it converges to zero in an infinite time. This error is insignificant in practical terms.

The resulting error is used to calculate the attitude reference trajectory shown before in section IV-.1, see Figure 3. In Figure 9 the translational velocity errors in a logarithmic scale are depicted, here it can be seen that the angular velocity converges to zero at an infinite time. In practical terms it can be considered that the system's angular velocity really converges to zero. Figure 10 shows the system inputs τ_{ux}, τ_{uy}, and τ_{uz}, notice that they converge to zero when the system stabilizes.

V. Test Bench

Our prototype is a quadcopter assembled at the UMI LAFMIA laboratory. This prototype is composed as follows: the structure is a carbon fiber frame. The four motors are 3525-650Kv 14-Pole Brushless Motors, they are controlled

using four 30 Ampere ESC Multi-rotor Motor Speed Controllers. The radio-transmitter has 9 PPM channels working 3.4 GHz. The battery is a 5000mAh 4-cell 14.8V Li-Po Battery. Also, the processor is a 32Bit Autonomous Vehicle Micro-computer, see Figure 11.

The processor used is based on the open hardware Pixhawk board, [13], which is an "all in one" unit that combines all the necessary sensors to stabilize the system in attitude, as well as several interfaces for other kinds of inputs and outputs. It has a 168 MHz Cortex M4F CPU with 156 KB RAM, and 2 MB Flash, it includes 3D accelerometers, gyroscopes, magnetometers, and barometers.

VI. Flight tests

A. Attitude test

The attitude control law (12) is implemented on the experimental platform. The references used came from a manual radio frequency controller in order to test the validity of the attitude control law.

The test was performed in an outdoor environment, which means that the quadcopter was subjected to a considerable amount of perturbations. It can be seen that, despite the interferences, the platform follows the trajectory. The quaternion error used has the same definition as the one used for the simulations. The control law stabilizes this error quaternion to one.

The feedback gain used in this control law was obtained empirically and has the following form:

$$K_{att} = \begin{bmatrix} 0.35 & 0 & 0 & 0.08 & 0 & 0 \\ 0 & 0.35 & 0 & 0 & 0.08 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0.2 \end{bmatrix},$$

which is implemented as seen in equation (12). It is important to note that this state feedback uses the axis-angle representation with the logarithmic mapping found in equation (5).

All information presented in Figures 12 – 15 was logged inside the vehicle's control board and later retrieved from its memory. The board can also be configured to broadcast this information immediately using a radio transmitter.

Figure 12 shows the reference quaternion given through the radio controller. It can be seen in the first part that there

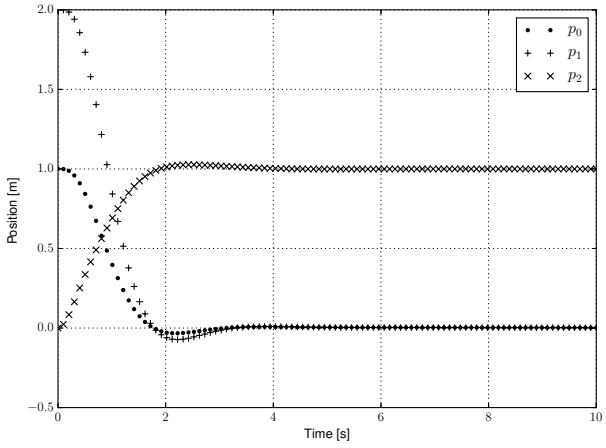

Fig. 6. Quadcopter translational position (p).

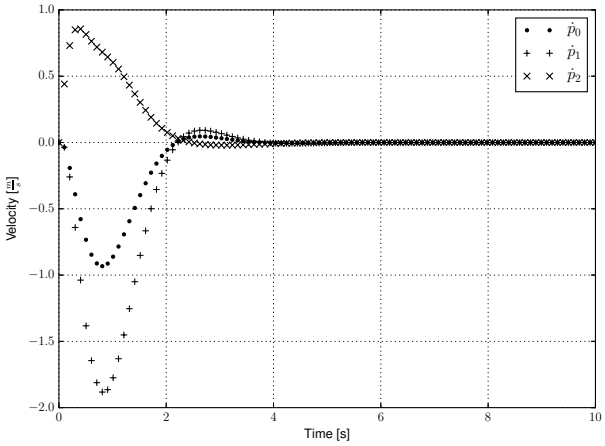

Fig. 7. Quadcopter translational velocity ($\dot{p}$).

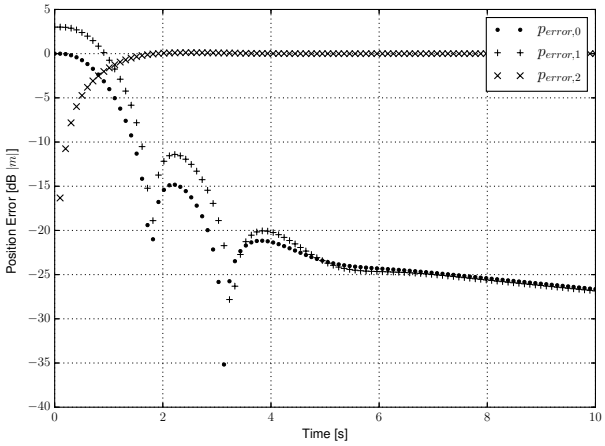

Fig. 8. Quadcopter translational position error.

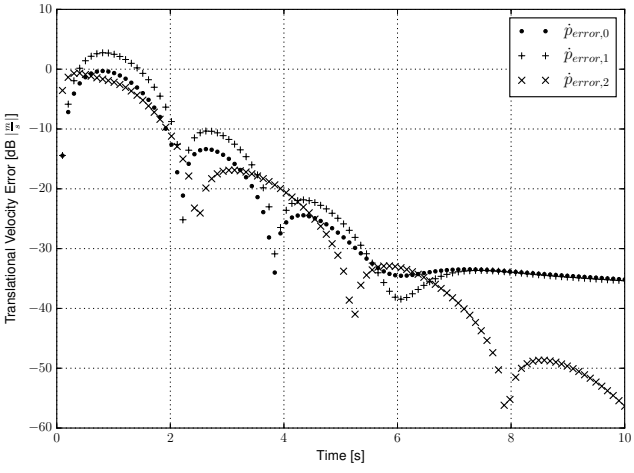

Fig. 9. Quadcopter translational velocity error.

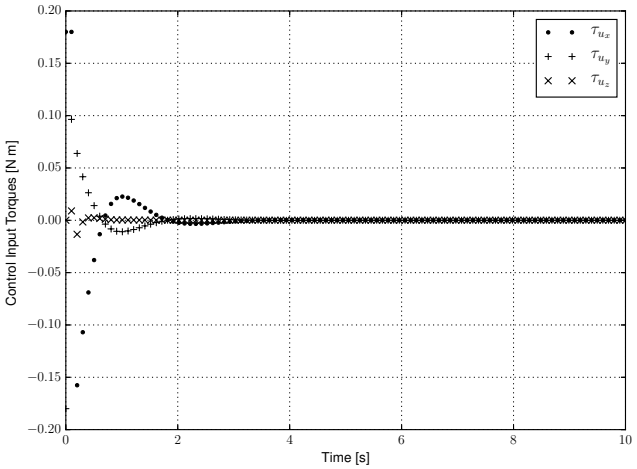

Fig. 10. Quadcopter input.

are just a few changes in order to correct the position of the vehicle according to the pilot's references. A change in the yaw angle of the vehicle can be seen near the end of the plot. Figure 13 shows the real attitude trajectory that the vehicle followed using the proposed control law. This information was obtained and filtered from the vehicle's IMU. It can be noted its similarity with the desired trajectory.

Figure 14 shows the quaternion error as described in the simulation section. The imaginary vector of this quaternion tends to stabilize around one. Figure 15 shows the angular velocity in rad/s. It can be seen that it is smooth and is also affected by the external disturbances. Since it is a stable system, the angular velocity tends to stay around zero.

B. Future Work

The stability of the attitude control was proven for a manual trajectory in an outdoor environment. In order to stabilize the full states, a translational trajectory must be fed to this control law in order to stabilize the translational part of the vehicle. In this test, a translational velocity reference will be given using the radio transmitter. The speed will be sensed with an optical flow sensor camera so that the vehicle stabilizes at the desired velocity.

Since the rotational and the translational model have an algebraic relationship given by equation (14), it is expected that the attitude system converges to the trajectory needed to stabilize towards the desired velocity.

VII. Conclusion

A quaternion model for the quadrotor vehicle is presented in this paper. This model was separated in two dynamics, one for the rotational and another for the translational stabilization. The dynamic model was simulated in Python programming language, and successfully stabilized using a PD controller. Although only the rotational control law was implemented in the experimental platform, the results from the simulations and from the flight tests let us foresee promising results for future work.

An exact linearization is also realized in order to apply linear control algorithms. Moreover, it was shown that the quadrotor system can be stabilized asymptotically using simple state feedback controls by taking advantage of the mathematical simpleness of the quaternion exponential mapping. The test showed that, even in outdoor environments, the attitude remains stable using the proposed control law.

In addition, the usage of quaternions also eliminates undesired effects on the platform such as the gimball-lock or discontinuities, which are common problems using traditional approaches. This makes possible to accomplish complex tasks and applications such as surveillance using gimbaled cameras or object manipulation.

References

[1] S. Spedicato et al. Aggressive maneuver regulation of a quadrotor UAV. In The 16th International Symposium on Robotics Research, Singapore, 2013

Fig. 11. Quadcopter platform

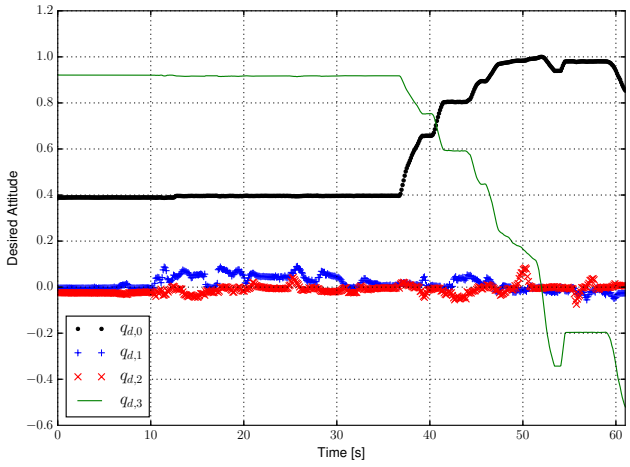

Fig. 12. Quadcopter reference attitude (q_d).

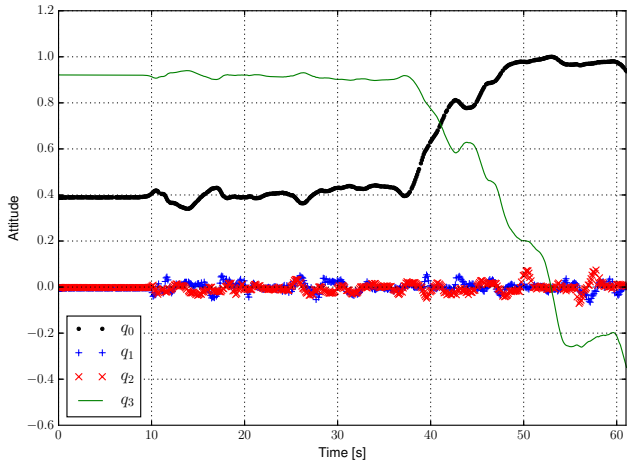

Fig. 13. Quadcopter attitude (q).

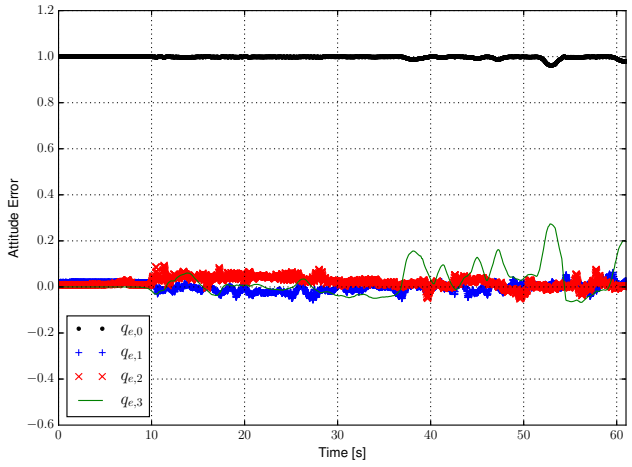

Fig. 14. Quadcopter attitude error (q_e).

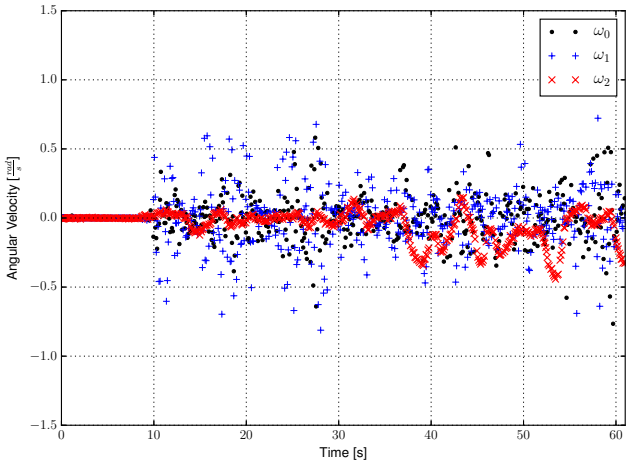

Fig. 15. Quadcopter rotational velocity ω.

[2] T. Lee et al. Geometric Tracking Control of a Quadrotor UAV on SE(3). In The 49th IEEE Conference on Decision and Control, Atlanta, GA, USA, 2010

[3] A Alaimo, V Artale, C Milazzo, A Ricciardello, and L Trefiletti. Mathematical modeling and control of a hexacopter. In Unmanned Aircraft Systems (ICUAS), 2013 International Conference on, pages 1043-1050. IEEE, 2013.

[4] Ricardo Campa and Karla Camarillo. Unit quaternions: A mathematical tool for modeling, path planning and control of robot manipulators. Robot manipulators, M. Ceccarelli (ed.), In-Teh, pages 21-48, 2008.

[5] Simon L Altmann. Hamilton, rodrigues, and the quater- nion scandal. Mathematics Magazine, pages 291-308, 1989.

[6] Kerry W Spring. Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: a review. Mechanism and Machine Theory, vol.21, no.5 pp. 365-373, 1986.

[7] Jack B Kuipers. Quaternions and rotation sequences, volume 66. Princeton university press Princeton, 1999.

[8] Xiangke Wang and Changbin Yu. Feedback lin- earization regulator with coupled attitude and translation dynamics based on unit dual quaternion. In Intelligent Control (ISIC), 2010 IEEE International Symposium on, pages 2380-2384. IEEE, 2010.

[9] Herbert Goldstein. Classical mechanics, volume 4. Pearson Education India, 1962.

[10] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL http://www.scipy.org/. [Online; accessed 2015-02-05].

[11] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. Computing in Science & Engineering, 13(2):22-30, 2011.

[12] James Goppert et al. Python control systems library, 2014. URL *http://sourceforge.net/projects/python-control/*. [Online; accessed 2015-02-05].

[13] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In Robotics and automation (ICRA), 2011 IEEE international conference on, pages 2992-2997. IEEE, 2011.

[14] John D Hunter. Matplotlib: A 2d graphics environment. Computing in science and engineering, 9(3):90-95, 2007.

[15] Emil Fresk and George Nikolakopou- los. Full quaternion based attitude control for a quadrotor. In Control Conference (ECC), 2013 European, pages 3864-3869. IEEE, 2013.

[16] Pedro Castillo Garcia, Rogelio Lozano, and Alejandro Enrique Dzul. Modelling and control of mini-flying machines. Springer Science & Business Media, 2006.

[17] Katsuhiko Ogata and Yanjuan Yang. Modern control engineering. 1970.

[18] Jean-Jacques E Slotine, Weiping Li, et al. Applied nonlinear control, volume 199. Prentice-Hall Englewood Cliffs, NJ, 1991.

[19] Kimon P Valavanis. Advances in unmanned aerial vehicles: state of the art and the road to autonomy, volume 33. Springer Science & Business Media, 2008.