Implementation of a quadrotor UAV

Santiago Paternain
Faculty of Engineering
University of the Republic
Montevideo, Uruguay
spaternain@gmail.com

Rodrigo Rosa
Faculty of Engineering
University of the Republic
Montevideo, Uruguay
rodrigorosa.LG@gmail.com

Matías Tailanián
Faculty of Engineering
University of the Republic
Montevideo, Uruguay
matias@tailanian.com

Rafael Canetti
Faculty of Engineering
University of the Republic
Montevideo, Uruguay
canetti@anii.org.uy

***Abstract*—This paper describes the design and integration of a control system that allows the autonomous flight of a commercial quadrotor. A mathematical model for the quadrotor is developed, its parameters determined from the characterization of the unit. An external intelligence is integrated to play the rol of the flight controller. A 9 degrees of freedom Inertial Measurement Unit (IMU) equipped with a barometer is calibrated and added to the platform. Data from the IMU is combined with the information provided by a GPS within a Extended Kalman Filter (EKF) to obtain a reliable estimation of the state variables. The control actions are obtained from a proportional-integral controller based on the LQR algorithm. For windless environments, a stable platform is achieved.**

I. Introduction

A quadrotor platform can be imagined performing countless tasks. Many applications are being developed based on such platforms, but how do they fly? A aerial vehicle is inherently unstable, staying still is not a simple task. This paper focuses on the stabilization of a quadrotor, explaining how to work with an IMU [1] [2] [3], the development of the mathematical model [4] [5] that represents the system, the filtering techniques applied to sensor data [6] [7], and the control system [4] [10] that allows the quadrotor to fly.

The platform is based on a commercial radio controlled quadrotor, the frame and the motor control system were preserved, the IMU and intelligence were replaced by the flight controller that was developed. A BeagleBoard[1] running Linux[2] performs the computations requiered to convert raw data received from the IMU[3] over a UART and combine it using an EKF, overcoming the problems inherent to each sensor and the noise generated by the vibrations of the motors, and providing a reliable estimation of the state vector. Once the current state is known, the LQR algorithm [8] [9] is used to derive the control actions required to bring the system to the desired setpoint.

II. Model of a quadrotor

A diagram of the quadrotor is shown in figure (1). Two of the motors rotate clockwise (2 and 4) and the other two (1 and 3) rotate clockwise. This configuration allows the quadrotor to rotate, tilt and gain/lose altitud by setting different speeds on each motor. Two frames of reference are constantly used through out this paper: An intertial frame S_I (relative to the Earth) $\{\hat{\imath}, \hat{\jmath}, \hat{k}\} - \{\text{North}, \text{West}, \text{Up}\})$ and a non-intertial frame S_q relative to the quadrotor. The mapping of one frame to the other can be achieved by applying the three rotations show in figure (2). The angles $\{\theta, \varphi, \psi\}$ are know as Euler angles.

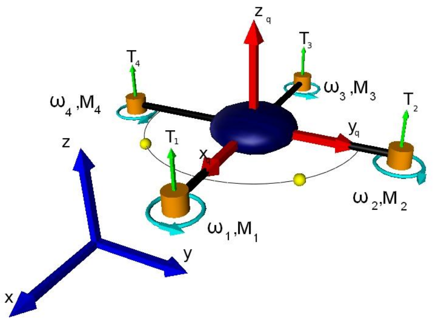
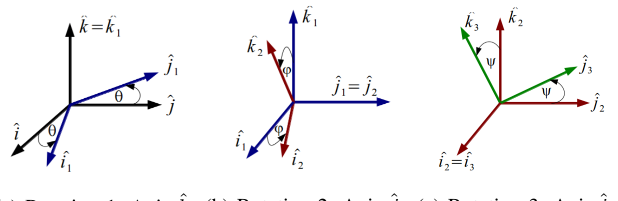


Fig. 1: **Model of the quadrotor -** The blue arrows represent the intertial reference frame $S_I - \{\hat{\imath}, \hat{\jmath}, \hat{k}\}$ $(\{\vec{x}, \vec{y}, \vec{z}\})$, mapped to North, West and Up respectively. The red arrows represent the non-inertial reference frame $S_q - \{\hat{\imath}_q, \hat{\jmath}_q, \hat{k}_q\}$ $(\{\vec{x}_q, \vec{y}_q, \vec{z}_q\})$ relative to the quadrotor. The cyan "looped" arrows indicate the direction of rotation of each motor (right hand rule applied to the green vectors labeled $T_{[1,2,3,4]}$). The semicircle and the two yellow spheres indicate the front of the unit.



(a) Rotation 1: Axis $\hat{k}$ (b) Rotation 2: Axis $\hat{\jmath}$ (c) Rotation 3: Axis $\hat{\imath}$

Fig. 2: Rotations applied on S_I to obtain S_q.

From a detailed analysis of the dynamics and kinematics of the quadrotor, the state vector $\vec{X}$ given by ((1)) is built to describe the system at any given time. Practical considerations lead to choosing the state variables referenced to the quadrotor frame S_q. This choice introduces simplifications in the theoretical development, and simplifies the interpretation of the data

[1]BeagleBoard development board - http://beagleboard.org/
[2]Angstrom distribution: http://www.angstrom-distribution.org/
[3]*Mongoose* IMU - http://store.ckdevices.com/

provided by the IMU, which is mounted on the quadrotor and hence provides provides accelerations and angular velocities that are relative to S_q:

$$\vec{X} = \{x, y, z, \theta, \varphi, \psi, v_{q_z}, v_{q_y}, v_{q_z}, \omega_{q_x}, \omega_{q_y}, \omega_{q_z}\} \tag{1}$$

where:

- $\{x, y, z\}$ are the cartesian coordinates in S_I.
- $\{\theta, \varphi, \psi\}$ are the Euler angles show in ((2)).
- $\{v_{q_x}, v_{q_y}, v_{q_z}\}$ are the linear velocities relative to S_I.
- $\{w_{q_x}, w_{q_y}, w_{q_z}\}$ are the angular velocities relative to S_q (right hand rule applied on $\{\hat{i}_q, \hat{j}_q, \hat{k}_q\}$).

The equations that govern the dynamics of the quadrotor, which are highly non-linear, are presented in equation (2).

$$\boxed{\begin{aligned}
&\dot{x} = v_{q_x}\cos\varphi\cos\theta + v_{q_y}(\cos\theta\sin\varphi\sin\psi - \cos\varphi\sin\theta)\\
&\quad + v_{q_z}(\sin\psi\sin\theta + \cos\psi\cos\theta\sin\varphi)\\
&\dot{y} = v_{q_x}\cos\varphi\sin\theta + v_{q_y}(\cos\psi\cos\theta + \sin\theta\sin\varphi\sin\psi)\\
&\quad + v_{q_z}(\cos\psi\sin\theta\sin\varphi - \cos\theta\sin\psi)\\
&\dot{z} = -v_{q_x}\sin\varphi + v_{q_y}\cos\varphi\sin\psi + v_{q_z}\cos\varphi\cos\psi\\
&\dot{\psi} = \omega_{q_x} + \omega_{q_z}\tan\varphi\cos\psi + \omega_{q_y}\tan\varphi\sin\psi\\
&\dot{\varphi} = \omega_{q_y}\cos\psi - \omega_{q_z}\sin\psi\\
&\dot{\theta} = \omega_{q_z}\frac{\cos\psi}{\cos\varphi} + \omega_{q_y}\frac{\sin\psi}{\cos\varphi}\\
&\dot{v_{q_x}} = v_{q_y}\omega_{q_z} - v_{q_z}\omega_{q_y} + g\sin\varphi\\
&\dot{v_{q_y}} = v_{q_z}\omega_{q_x} - v_{q_x}\omega_{q_z} - g\cos\varphi\sin\psi\\
&\dot{v_{q_z}} = v_{q_x}\omega_{q_y} - v_{q_y}\omega_{q_x} - g\cos\varphi\cos\psi + \frac{1}{M}\sum_{i=1}^{4} T_i\\
&\dot{\omega_{q_x}} = \frac{1}{I_{xx}}\omega_{q_y}\omega_{q_z}(I_{yy} - I_{zz})\\
&\quad + \frac{1}{I_{xx}}\omega_{q_y} I_{zz_m}(\omega_1 - \omega_2 + \omega_3 - \omega_4)\\
&\quad - \frac{1}{I_{xx}} dMg\cos\varphi\sin\psi + \frac{1}{I_{xx}} L(T_2 - T_4)\\
&\dot{\omega_{q_y}} = \frac{1}{I_{yy}}\omega_{q_x}\omega_{q_z}(-I_{xx} + I_{zz})\\
&\quad + \frac{1}{I_{yy}}\omega_{q_x} I_{zz_m}(\omega_1 - \omega_2 + \omega_3 - \omega_4)\\
&\quad - \frac{1}{I_{yy}} dMg\sin\varphi + \frac{1}{I_{yy}} L(T_3 - T_1)\\
&\dot{\omega_{q_z}} = \frac{1}{I_{zz}}(-Q_1 + Q_2 - Q_3 + Q_4)
\end{aligned}} \tag{2}$$

To simplifly the control system, equations (2) are linearized near certain points of operation which result in a Linear Time Invariant (LTI) system. These points allow the following trajectories: Hovering[4], uniform linear trajectories, and uniform circular trajectories. The system will be restricted to these three trajectories, since it greatly simplifies the control system and does it not introduce significant limitations.

III. Kalman Filter

In order to perform adequate control actions, a reliable estimation of the state variables must be performed, in real time. The Kalman Filter uses the mathematical model for the system to predict what should happen next given the current state, and corrects the prediction with the information read from the sensors, taking into consideration how much confidence is placed on the prediction and how much on the measurements. This weighted prediction-correction technique allows a smooth state estimation without the typical delay introduced by filtering, even small delays can severly affect the performance of the system.

[4]Constant state vector.

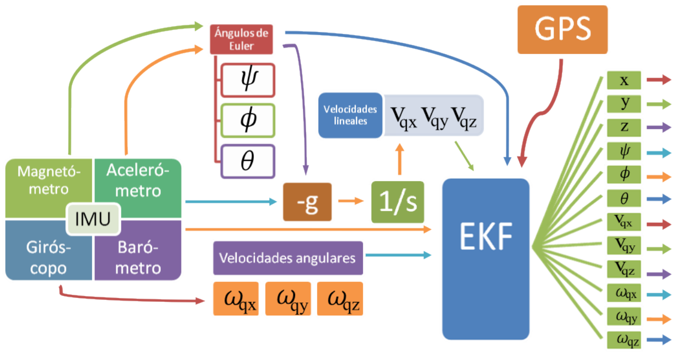


Fig. 3: **EKF -** Outline of how sensor data is combined to estimate the state variables.

Every sensor has its issues: The gyro drifts over time; the accelerometer is very sensible to the vibrations generated by the motors; the magnetometer is distorted by ferromagnetic materials; the GPS has very little precision and a slow update rate. Each sensor by itself is very limited, but they can be combined to compensate for their limitations. The filter takes care of this.

Two different types of data will be available, depending on the availability of GPS information. When no GPS data is available there will be no direct feedback from the sensors to correct the position on the horizontal plane, only through integration, so only the prediction step can be performed for these variables. As shown in figure (4), the estimation will drift until GPS data is available.

The theory behind a standard Kalman Filter does not hold for a system that is not linear. The model for the quadrotor given by (2) is highly non-linear, so an EKF is implemented based on [6] and [7]. An EKF is compatible with non-linear systems, but it is not optimal, the performance is highly dependant on the linearizacion [11]. The implementation of the EKF gave very good results, and proved to play a critical part in the system. As an example, the data provided by the accelerometer is "unusable" without filtering, and experiments with a simple low pass filter (LPF) showed that a 60ms delay introduced by the LPF severly deteriorated the performance of the system. When the EKF was assigned the task of reducing noise, the perfomance was significatively improved.

IV. Control design

V. Software

The software runs on several independant boards: An ARM-Cortex-A8 on the flight controller, an ATMega328p on the

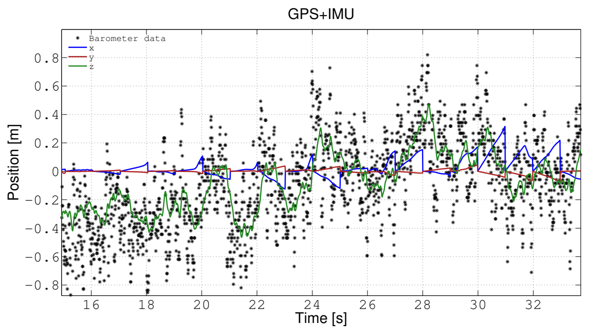


Fig. 4

IMU, and four microprocessors[5] on the electronic speed controller (ESC), one for each motor. The code running on the ESCs is not available, but the communication protocol was decoded using a logic analyzer. The firmware on the IMU takes care of reading from all the sensors (using I^2C) and sending a new frame of data every 10ms over a UART. The code running on the flight controller processes the data from the IMU, calculates the necessary control actions, and sets the desired speed for each motor by sending the appropriate commands, via I^2C, to the ESCs. It requires Linux specific C functions to handle I/O. It was developed in modules, replacing any part of it should be a straightforward task. The most critical aspect of the flight controller is timing, any delays will severely degrade the performance of the system.

VI. Flight tests

Many flight tests were performed, most of them with strings attached above and below the quadrotor as security measures. resultados buenos para lugares sin viento, posibles explicaciones

A. *Subsection Heading Here*

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

VII. Conclusion

The conclusion goes here.

A. *Future Work*

- The LotusRC ESCs proved to be extremely unreliable[6]. Random glitches cause the motors to turn off, which inevitably leads to an accident. Replacing the ESCs is a critical task.
- The choice of implementing the flight controller on an operating system that does not run in real-time made timing a complicated issue. These issues were dealt with, but assinging additional tasks to the system will surely distrupt timing. Implementing the stabilization system, which required a very fast and steady loop, on a dedicated microprocessor would be a better approach. This would allow the code running on top of the operating system to perform higher level tasks, such as Visual Simultaneous localization and mapping (VSLAM), navigation, tracking, etc.
- The fact that the control system does not work in a windy environment severely limits the range of possible applications. Modifying the control system to allow flight under windy conditions would be a significant improvement.

[5]C8051F330/1 - Silicon Labs.

[6]Several versions were tested, all of then showed the same issues.

Acknowledgment

The authors would like to thank de panader.

References

[1] I. Skog and P. Hndel, “Calibration of a mems inertial measurment unit,” in *XVII Imeko World Congress, Metrology for a Sustainable Development*, (Rio de Janeiro), Setiembre 2006.

[2] C. Konvalin, “Compensating for tilt, hard iron and soft iron effects,” Tech. Rep. MTD-0802, Memsense, Agosto 2008.

[3] A. Barraud, “Magnetometers calibration.” http://www.mathworks.com/matlabcentral/fileexchange/23398-magnetometers-calibration, Marzo 2009.

[4] T. Bresciani, “Modeling, identification and control of a quadrotor helicopter,” Master’s thesis, Lund University, Octubre 2008.

[5] M. Vendittelli, “Quadrotor modeling.” Curso: “Elective in robotics”, Sapienza Universit Di Roma, Noviembre 2011.

[6] H. Zhao and Z. Wang, “Motion measurement using inertial sensors, ultrasonic sensors, and magnetometers with extended kalman filter for data fusion,” *Sensors Journal, IEEE*, vol. 12, pp. 943–953, Mayo 2012.

[7] L. Tams, G. Lazea, R. Robotin, C. Marcu, S. Herle, and Z. Szekely, “State estimation based on kalman filtering techniques in navigation,” in *IEEE International Conference on Automation, Quality and Testing, Robotics*, pp. 147–152, IEEE, Mayo 2008.

[8] J. P.Hespanha, “Undergraduate lecture notes on lqg/lqr controller design,” 2007.

[9] A. F. Soslashrensen, “Autonomous control of a miniature quadrotor following fast trajectories,” Master’s thesis, Aalborg University/U.C. Berkeley, 2009-2010.

[10] M. Hakas, “Sistemas de control en tiempo discreto.” Curso: “Introducción a la Teoría de Control”, Facultad de Ingeniería Universidad de la República, 2005.

[11] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, vol. 1 of *Prentice Hall singal processing*. Pearson, 2 ed., 2011.