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 radio controlled commercial quadcopter. A mathematical model for the quadcopter 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 every 10ms. 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 quadcopter 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 quadcopter, explaining how to work with an IMU, the development of the mathematical model that represents the system, the filtering techniques applied to sensor data, and the control system that will allows to quadcopter to fly.

The platform is based on a commercial quadcopter, the frame and the motor control system were preserved, but the IMU and intelligence were replaced by the flight controller that was developed. An ARM-Cortex-A8 processor running Linux performs the computations requiered to convert raw IMU data and combine it using an EKF, overcoming the noise generated by the vibrations of the motors, and providing a reliable estimation of the state variables. Once the current state is known, the LQR algorithm is used to derive control actions required to bring the system to the desired setpoint.

A diagram of the quadcopter 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 quadcopter to rotate, tilt and gain/lose altitud.

II. Model of a quadrotor

The equations that govern the dynamics of a quadcopter are bla.

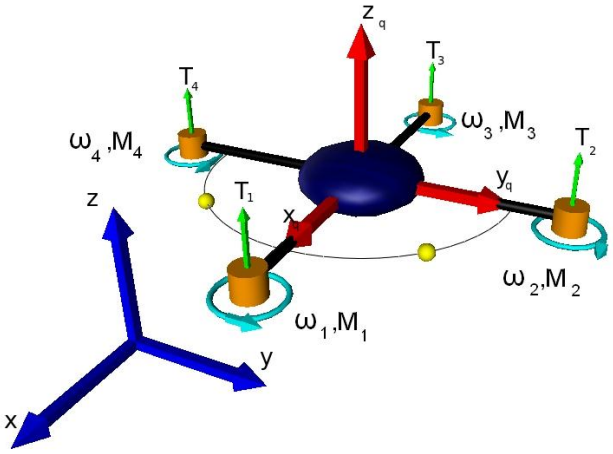

Fig. 1. Model of a quadcopter

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

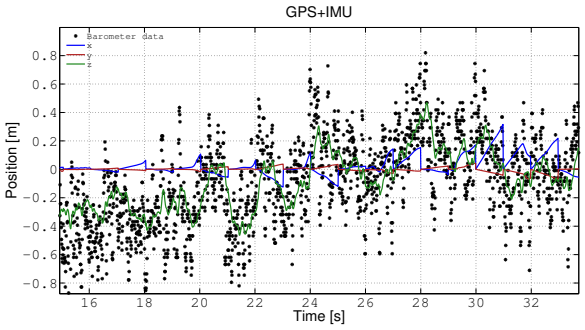

Fig. 2.

para hacerlo rotar para un lado hay q hacer esto, para el otro lo otro

integradores -¿ permiten tomar en consideracion cosas q le cuelges, desequilibrios

III. Kalman Filter

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

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.

The theory behind a standard Kalman Filter does not hold for a system that is not linear. The model for the quadcopter is highly non-linear, so an EKF is implemented based on [1] and [2]. An EKF is not optimal, and the performance is highly dependant on the linearizacion [3]. The performance achieved is considered acceptable.

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 2, the estimation will drift until GPS data is available.

IV. Control design

V. Software

The software is spread across several boards: An ARM-Cortex-A8 on the flight controller, an ATMega328p on the IMU, and four microprocessors[1] on the ESCs, 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.

The code 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

A. *Subsection Heading Here*

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

VII. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank de panader.

References

[1] 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.

[2] 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.

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

[1] C8051F330/1 - Silicon Labs.