CENTRALIZED MULTIVARIABLE LQG CONTROL SYSTEM FOR LONGITUDINAL AND LATERAL SPEED HOLD AUTOPILOT FOR THE AR.DRONE 2.0 QUADCOPTER

Antonio S. Silveira*, Anderson F. Silva†, José Augusto F. Real*, Orlando F. Silva*

*Laboratory of Controls and Systems at UFPA Institute of Technology
Belém, PA, Brazil

†Federal Institute of Education, Science and Technology of Pará
Parauapebas, PA, Brazil

Emails: asilveira@ufpa.br, anderson.silva@ifpa.edu.br, real@ufpa.br, orfosi@ufpa.br

Abstract— In this work it is presented the design procedures and experimental results of a centralized multivariable LQG control system for longitudinal and lateral speed hold autopilot for the AR.Drone 2.0 quadcopter. The main contribution is that instead of assuming that the longitudinal and lateral dynamics are completely decoupled, the quadcopter is modeled as a coupled multivariable state-space system with transport time-delay. The system identification procedure, by extended recursive least-squares estimation, is done directly in the state-space form and a detailed description of the equations derived for this project is given. The LQG design is aided by analysis on the system's step-response tests in the time domain and is based on a non orthodox Kalman filter design, dual to the LQR. The proposed speed hold autopilot is evaluated using the system's model and then applied to the real process. The experimental platform used was a free add-on toolbox for Matlab/Simulink, also used as a benchmark control system with a ready-to-fly example of decentralized proportional control system for the AR.Drone 2.0. Results summarized in a table of integral performance indexes and a discussion over the formalism of the LQG method and its applications to flight control systems concludes the contributions of this work.

Keywords— Speed hold autopilot, Linear Quadratic Gaussian, Linear Quadratic Regulator, Kalman Filter, AR.Drone 2.0.

1 Introduction

In this paper it is presented the design procedures and experimental results of a centralized multivariable linear quadratic Gaussian (LQG) control system for the longitudinal and lateral speed hold autopilot of a quadcopter, the Parrot's AR.Drone 2.0, which is a low cost unmanned aerial vehicle (UAV) developed and sold by Parrot.com. Moreover in this text the AR.Drone 2.0 will be addressed simply as the *AR.Drone* and *drone* for quadcopters in general.

To the best of the authors' knowledge, what is uniquely investigated in this paper is the use of an optimal centralized multivariable control design technique on the AR.Drone, whereas decentralized and proportional-integral-derivative (PID) control techniques are dominant among the automatic flight control systems used with quadcopters. In other words, at least for the AR.Drone, the standard procedure is to divide the multiple-input, multiple-output (MIMO) system into several single-input, single-output (SISO) subsystems assuming its dynamics to be decoupled, and then, successive control-loops may be closed forming a complex sub-optimal cascaded and decentralized control system.

One of the disadvantages of decentralized control in flight control systems is that every control-loop affects the other, and just after tuning an altitude hold autopilot, the longitudinal speed hold autopilot may disturb the former control-loop and vice-versa, sometimes giving rise to oscillatory behavior or even instability. Optimal model-based centralized methods, in contrast, solves the problem at once for all the variables involved in the problem, in a way that part of or all subsystems could be systematically decoupled using Modern Control Theory multivariable loop-shaping design techniques, such as LQG design based on principal gains analysis (Stevens et al., 2016).

The LQG system, i.e. the dynamic compensator comprised of a linear quadratic regulator (LQR) and the Kalman filter estimator (KF), is the theoretical base of some model-based advanced linear control techniques, e.g. Generalized Predictive Control (GPC), shown in Bitmead et al. (1990) to be a particular solution of the LQG. These techniques have in common the optimization of a cost function and, explicitly or implicitly, an observable/estimation part and an estimation-dependent or predictive-dependent feedback control part. These systems are far more complex than simple PID loops but they are extremely necessary for multivariable systems highly dependent on data fusion and data estimation, such as flight control systems, in which not every variable is measured and some are measured by multiple sensors, for redundancy and safety in case of fault or malfunction.

The study of complex control and systems problems such as those of guidance, navigation and control (GNC) of aerospace systems once were restricted to very few places in the world, at least for experimental tests due to the difficulties to acquire the hardware, the airframe, to have the place

and people to conduct flight tests. However, small UAVs have allowed many universities, even at developing countries, to get to known and to face real problems of 6-DOF (six degrees-of-freedom) aerial robotics. This is exactly one of the opportunities that just recently opened-up at the Laboratory of Control and Systems at UFPA Institute of Technology, after receiving the financial support of the Brazilian Research Agency, CNPq, to start a GNC research using quadcopters.

Since it is a late start in the study of applied control systems for quadcopters, if compared to other institutions in and out of Brazil – e.g., researches from the state of Espírito Santo with an earlier start, such as in Santos et al. (2014) and Santos et al. (2017) – this study began by looking into the history of flight control systems and its relation to the history of the Control Theory itself. Then, this information merged with the information of recent papers on quadcopters autopilots, it can be observed astonishing researches on applications with the drones, but whereas its control systems are far bellow acceptable if compared to the standards of the aerospace industry, as those used by Boeing and Lockheed, for example (Stevens et al., 2016). Of course that the *lack of control* or a bad autopilot is going to deteriorate the performance of the quadcopter to a specific application.

An example of how the AR.Drone's control system can compromise the application performance is remarked at the Discussions Section in the paper of Clark et al. (2017). The AR.Drone was used for autonomous remote visual inspection to generate a three-dimensional surface-meshed model of a nuclear intermediate level waste storage drum. The authors used distributed and scalable proportional controllers and concluded that their application results would be better if proportional-derivative (PD) controllers have been used instead.

The AR.Drone has been used within the academic scenario with more sophisticated control approaches, but has not being treated, to the best of the authors' knowledge, as a coupled MIMO system. For example, Armah and Yi (2015), worked with a PD altitude controller tuned based on a time-delayed system model; Hernandez et al. (2014) implemented optimal MPCs to the attitude, altitude and yaw problems and compared the results to PD controllers and despite outperforming the PD implementation, the design was still based on decentralized controllers, assuming the Drone as a decoupled airframe, controlling it as four independent SISO systems; Criado and Rubio (2015) worked with the Drone for autonomous path tracking control using PID and GPC, and for both cases a cascaded control structure was used, being the inner loop the speed control loop using non-linear saturation as an antiwindup method and the outer loop a position control loop. Their speed control loop had no control augmentation system (CAS) (Stevens et al., 2016), counting on a separate non-linear saturation rule to prevent the integrator wind-up phenomenon.

Considering the presented short review and the lack of centralized multivariable control applied to the AR.Drone, the objective is the design of a single MIMO LQG system for

- the lateral **speed hold autopilot**
  - with integrated **roll rate CAS**
- the longitudinal **speed hold autopilot**
  - with integrated **pitch rate CAS**

This control system design should suffice to point out the benefits of using LQG-like methods in the centralized MIMO form for quadcopters. Also, this simple autopilot with integrated CAS systems would encourage the search for complete flight control systems, from the stability augmentation systems (SAS) to the autopilots, for drones which a standard GNC structure is not already defined by the aerospace industry.

All systems models within this paper are based on discrete linear time-invariant systems and beyond this introductory part, this work is organized in the following form: Section 2 is used to present the control problem, system's I/Os and state variables. SAS, CAS and autopilots are also defined to give some formalism with respect to the flight control systems industry. In Section 3 it is explained how to use the recursive least-squares estimator for state-space modeling, describing a powerful identification tool to develop complex state-space systems and adaptive state-space systems. Section 4 covers the LQG design, as traditional as it is, all needed steps are shown in a straightforward manner. In Section 5, a brief description of the experimental setup for flight tests, along with its software and hardware used, is given. This section is finished with the presentation of the main experimental results and is followed by the Conclusions.

2 Problem formulation

Following the standards of aircraft practice, the AR.Drone orientation is given in the x, y, z axes, in which the usual reference system on Earth is the *ned* system, acronym for North-East-down, that defines positive directions for the x-axis pointing (*true*) North, y-axis pointing East and z-axis pointing down. Similarly, considering the quadcopter body-fixed reference frame, the *frd* system – acronym for *forward* (x-axis or longitudinal axis), *right* (y-axis or lateral axis), *down* (z-axis) – is considered. According to Stevens et al. (2016),

- Right-handed rotation about the x-axis gives positive ϕ, the roll angle.
- Right-handed rotation about the y-axis gives positive θ, the pitch angle.
- Right-handed rotation about the z-axis gives positive ψ, the compass heading (yaw) angle.

The ϕ, θ, ψ are known as the Euler Angles and they measure the attitude of the airframe with respect to the inertial *ned* frame. For quadcopters, changes to ϕ, θ angles can be used to generate lateral and longitudinal speeds, respectively. In this way, *ideally*, a drone can stay at a fixed heading and change its lateral and longitudinal positions through roll and pitch changes or even hold its position if its speeds are null. This is the basic concept behind a Speed Hold autopilot, described next among other types of flight control systems.

Automatic flight control systems and functions can be divided into three categories (Stevens et al., 2016):

- stability augmentation systems, SAS: e.g. roll damper, pitch damper, yaw damper.
- control augmentation systems, CAS: e.g. roll rate, pitch rate.
- autopilot functions: e.g. speed hold, pitch-attitude hold, roll angle hold, heading hold, altitude hold.

The AR.Drone is a ready-to-fly quadcopter with a basic stabilizer or SAS for roll, pitch and yaw damping, along with the autopilot function for altitude hold. As a ready-to-fly solution, this drone comes with a ready to use navigation system and a complete description is given by Mac et al. (2018). Within the present work, however, only the following states are used, shown within the discrete state vector $\mathbf{x}(k)$:

$$\mathbf{x}(k) = \begin{bmatrix} v(k) & u_{vel}(k) & \phi(k) & \theta(k) \end{bmatrix}^T, \quad (1)$$

where $v(k), u_{vel}(k)$ are, respectively, the lateral speed and longitudinal speed.

For the speed hold autopilot, the control problem is to control $v(k), u_{vel}(k)$ in order to follow a 2×1 reference speed vector $\mathbf{y_r}(k)$, preferably asymptotically and without any steady-state error. Since to change speeds implicate in changes in the roll and pitch angles, it is also necessary to compensate this variables somehow. In the majority of works done with the AR.Drone, it is generally applied an inner loop design for the roll CAS and pitch CAS and then the speed hold controller comes cascaded as an outer loop. However, with LQG design and with the state vector shown, this can be done optimally at once and this is the problem to be solved in this work.

In order to design the LQG system for the speed hold autopilot with integrated roll rate CAS and pitch rate CAS, the following design model will be used:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{A}\mathbf{x}(k-1) + \mathbf{B}\mathbf{u}(k-d) + \mathbf{G}\xi(k-1) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \nu(k) \end{aligned} \quad (2)$$

This state-space model is d-samples delayed with 2-input, 2-output, 4-state variables, linear time invariant system defined in the discrete-time domain, where $\mathbf{u}(k)$, $\mathbf{y}(k)$, $\xi(k)$, $\nu(k)$ are, respectively, the inputs vector, the outputs vector, the 2×1 process and outputs white Gaussian disturbances vectors which are related to the model uncertainties and measurement noise, respectively.

Similar to the controls of an airplane, a drone can change its roll and pitch angles through change in momentum as if having *ailerons* at its wings and *elevator* at its tail, but in fact this commands are produced by changes in motor torques as described, for the AR.Drone, in Mac et al. (2018). For the present work problem formulation, all that is needed to be known is that the 1st input modifies the roll angle and the 2nd modifies the pitch angle and throughout this text this inputs are addressed as *ailerons* and *elevator* commands.

The outputs of the system, $\mathbf{y}(k)$, are the measurements of the states $v(k), u_{vel}(k)$, which are available from the navigation system of the AR.Drone. Due to this, it must be remarked that $\phi(k), \theta(k)$ are estimated by the Kalman filter system within the LQG. However, it must be remarked that the AR.Drone's $\phi(k)$ and $\theta(k)$ angles are available for feedback, but within this paper a more complicated scenario for state estimation is going to be covered, considering a malfunction or the absence of these sensors.

Since the outputs vector is already defined, the $\mathbf{C}$ matrix is readily known as being

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (3)$$

The $\mathbf{A}, \mathbf{B}, \mathbf{G}$ matrices, however, are going to be estimated from recorded flight data as shown in the next section.

3 State-space system identification

The state-space RLS estimator algorithm can be further investigated in Malik (2004), but in this work it is used essentially in the same manner employed for SISO Auto-Regressive models identification, in the extended RLS case. Considering the 4-state variables model shown at (2), four RLS estimation problems can be solved with respect to $x_i(k), i = 1, 2, 3, 4$ such that the estimated $\hat{x}_i(k)$,

where the hat ˆ denotes estimated signals and parameters, is given by

$$\hat{x}_i(k) = \underbrace{\begin{bmatrix} \mathbf{x}(k-1) \\ \mathbf{u}(k-d) \\ \xi(k-1) \end{bmatrix}^T}_{\phi_{rls}} \underbrace{\begin{bmatrix} \hat{\mathbf{A}}_{i\times 4} \\ \hat{\mathbf{B}}_{i\times 2} \\ \hat{\mathbf{G}}_{i\times 2} \end{bmatrix}}_{\hat{\theta}_i} \tag{4}$$

By using the logged flight data and the estimated states models shown in (4), the RLS algorithm is implemented as it follows:

$$\mathbf{L}_i(k) = \frac{\mathbf{P}_i(k-1)\phi_{rls}(k)}{\left[1 + {\phi_{rls}}^T(k)\mathbf{P}_i(k-1)\phi_{rls}(k)\right]} \tag{5}$$

$$\hat{\theta}_i(k) = \hat{\theta}_i(k-1) + \mathbf{L}_i\left[x_i(k) - {\phi_{rls}}^T(k)\hat{\theta}_i(k-1)\right] \tag{6}$$

$$\mathbf{P}_i(k) = \left[\mathbf{I}_{8\times 8} - \mathbf{L}_i(k){\phi_{rls}}^T(k)\right]\mathbf{P}_i(k-1) \tag{7}$$

$$\xi(k) = \mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}(k) \tag{8}$$

The identified AR.Drone model matrices, obtained from logged flight data, after successive flight runs, in closed-loop with the LQG Speed Hold autopilot, for offline retuning, are given by

$$\mathbf{A} = \begin{bmatrix} 0.9949 & -0.0227 & 0.5332 & 0.0528 \\ 0.0340 & 0.9392 & 0.1426 & -0.4166 \\ -0.0060 & 0.0105 & 0.8496 & -0.0148 \\ -0.0038 & 0.0080 & -0.0106 & 0.7924 \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 0.0917 & -0.0216 \\ 0.0202 & -0.0262 \\ 0.0609 & 0.0275 \\ 0.0371 & 0.0466 \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 0.0074 & 0.0272 \\ -0.0490 & 0.0488 \\ 0.0071 & -0.0080 \\ 0.0008 & 0.0021 \end{bmatrix} \tag{9}$$

The sampling time used to register the flight data and also used with all flight control systems in this work is $T_s = 65ms$. Also, by analysis of the flight data, the discrete time-delay $d = 6$ was observed along with a Gaussian disturbance variance for $\xi(k) = \begin{bmatrix} \xi_1(k) & \xi_2(k) \end{bmatrix}^T$ respectively being ${\sigma_\xi}^2 = \begin{bmatrix} 0.0083 & 0.0041 \end{bmatrix}$.

It must be remarked that the focus of this paper is in no way the system identification procedure. Due to this, there is no need to present the validation procedure (nor to consider tests with another parameter estimation technique) but simply to state that it has worked sufficiently enough, considering the Parsimony Principle, reflecting the process dynamics within the operational flight condition of all experiments conducted in this work. And also, to cope with this information, this work is not based solely on simulations, but it bridges theory and practice, by applying every concept of the presented theory to a 6-DOF UAV, under lack of inertial measurement unit data (no roll and pitch angle sensors), transport time delays and coupled lateral and longitudinal dynamics.

4 LQG system design

Autopilot functions generally requires the flight system to be *hold* onto some desired operational flight condition. Specifically, the Speed Hold can be used to track and maintain a speed value or regulate the speeds at zero for quadcopters. In order to have a LQG system to be used for both reference tracking and regulation, incremental control or the velocity form control algorithm must be used, in order to have an offset free response to step-like constant references. It means that the input to the Drone is

$$\mathbf{u}(k) = \mathbf{u}(k-1) + \Delta\mathbf{u}(k), \quad \Delta = \left(1 - z^{-1}\right) \tag{10}$$

where $\Delta\mathbf{u}(k)$ is the discrete difference of the control input.

The LQG system design model is then the augmented by *integrators* version of the system's model, which is constructed on the basis of the augmented state vector $\mathbf{x_a}(k) = \begin{bmatrix} \Delta\mathbf{x}(k) & \mathbf{y}(k) \end{bmatrix}^T$, leading to

$$\begin{aligned} \mathbf{x_a}(k) &= \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{0}_{4\times 2} \\ \mathbf{CA} & \mathbf{I}_{2\times 2} \end{bmatrix}}_{\mathbf{A_a}} \mathbf{x_a}(k-1) + \underbrace{\begin{bmatrix} \mathbf{BP} \\ \mathbf{CBP} \end{bmatrix}}_{\mathbf{B_a}} \Delta\mathbf{u}(k-1) \\ \mathbf{y_a}(k) &= \underbrace{\begin{bmatrix} \mathbf{0}_{2\times 4} & \mathbf{I}_{2\times 2} \end{bmatrix}}_{\mathbf{C_a}} \mathbf{x_a}(k) \end{aligned} \tag{11}$$

where $\mathbf{P}$ is the inverse of the open-loop system DC gain, $\mathbf{P}^{-1} = \mathbf{C}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$. Then $\mathbf{P}$ is a 2×2 DC precompensator that is used to compensate the low frequency coupling as suggested in the Chapter 5 of Stevens et al. (2016), respective to multivariable frequency-domain techniques. It must be remarked that in the design model the total time delay of $d = 6$ and the Gaussian disturbance $\xi(k)$ are not considered, but the design will be tested onto the simulation model with noise and delay as presented in (2).

The complete LQG system is shown in the block diagram of Figure 1, comprised of the Kalman estimator, the LQR's gains and how they connect to the AR.Drone's simulation model shown in (2).

4.1 Linear quadratic regulator design

The design of the discrete full state-feedback LQR system can be found in several books on the subject of Modern Control Theory. Due to this, it is briefly presented as a step-by-step design algorithm. First of all, it is assumed that the process to be controlled is controllable and all its states are available to be fed back. Specifically to the AR.Drone's augmented model, the system is in

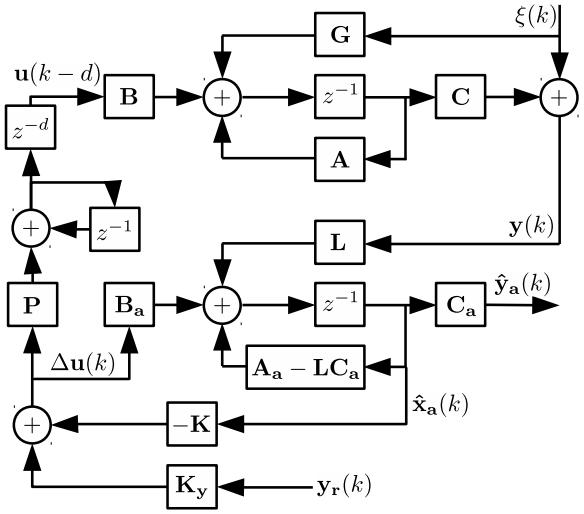

Figure 1: Block diagram of the LQG system.

fact controllable, but only $x_1(k), x_2(k)$ are available, but all states will later be available by state estimation usgin the Kalman filter.

The LQR problem, in the infinite horizon and discrete case for the augmented model, consists on minimizing the following quadratic performance index (Bitmead et al., 1990; Stevens et al., 2016):

$$J = \sum_{k=0}^{\infty} \left[\mathbf{x_a}^T Q_{lq} \mathbf{x_a} + \Delta\mathbf{u}^T R_{lq} \Delta\mathbf{u} \right] \tag{12}$$

where Q_{lq} is a symmetric positive-semidefinite weighting matrix for the states minimization and R_{lq} is a positive definite symmetric weighting matrix for the minimization of the control increment $\Delta\mathbf{u}(k)$.

The solution of the LQR problem leads to the computation of the optimal full state-feedback gain $\mathbf{K}$ that minimizes J when $\Delta\mathbf{u}(k) = -\mathbf{K}\mathbf{x_a}(k)$ is applied to the input of the augmented model shown in (11) and $\mathbf{x_a}(k) = (\mathbf{A_a} - \mathbf{B_a}\mathbf{K})\mathbf{x_a}(k-1)$. The computation of $\mathbf{K}$ is done by

$$\mathbf{K} = \left[\mathbf{A_a}^T \mathbf{P_{lq}}(\infty) \mathbf{B_a} (\mathbf{B_a}^T \mathbf{P_{lq}}(\infty) \mathbf{B_a} + R_{lq})^{-1} \right]^T \tag{13}$$

where $\mathbf{P_{lq}}(\infty)$ is solved by the controller case of the Riccati Difference Equation, when $k \to \infty$:

$$\mathbf{P_{lq}}(k+1) = \mathbf{A_a} \mathbf{P_{lq}}(k) \mathbf{A_a} - \mathbf{A_a}^T \mathbf{P_{lq}}(k) \mathbf{B_a} (\mathbf{B_a}^T \mathbf{P_{lq}}(k) \mathbf{B_a}^T + R_{lq})^{-1} \mathbf{B_a}^T \mathbf{P_{lq}}(k) \mathbf{A_a} + Q_{lq} \tag{14}$$

Since the reference tracking case needs to be covered by this LQR design, the control increment law changes to (Bitmead et al., 1990):

$$\Delta\mathbf{u}(k) = -\mathbf{K}\mathbf{x_a}(k) + \mathbf{K_y}\mathbf{y_r}(k) \tag{15}$$

where $\mathbf{K_y}$ corresponds to the columns of $\mathbf{K}$ that affects $\mathbf{y}(k)$ in the augmented state vector $\mathbf{x_a}(k) = \begin{bmatrix} \Delta\mathbf{x}(k) & \mathbf{y}(k) \end{bmatrix}^T$, which are the last two columns in this MIMO case with 2-outputs.

For the AR.Drone Speed Hold controller, the following weighting matrices were selected by trial and error while evaluating the unit-step response and Principal Gains in the frequency domain:

$$\begin{aligned} Q_{lq} &= diag \begin{pmatrix} 1 & 1 & 10^3 & 10^3 & 10^{-2} & 10^{-2} \end{pmatrix} \\ R_{lq} &= diag \begin{pmatrix} 50 & 50 \end{pmatrix} \end{aligned} \tag{16}$$

This selection led to the following $\mathbf{K}$, $\mathbf{K_y}$:

$$\begin{aligned} \mathbf{K} &= \begin{bmatrix} 0.5786 & 0.1232 & 2.5691 & 0.3613 & 0.0139 & -0.0004 \\ 0.3136 & 0.2263 & 0.9031 & -1.4670 & 0.0008 & 0.0137 \end{bmatrix} \\ \mathbf{K_y} &= \begin{bmatrix} 0.0139 & -0.0004 \\ 0.0008 & 0.0137 \end{bmatrix} \end{aligned} \tag{17}$$

4.2 Kalman filter design

The Kalman filter estimator used in this work, also known as the output injection estimator, is designed based on a non orthodox method, which is *dual* to the LQR problem, already described in this work and that allows to pose the estimator quadratic performance index, particularized to the augmented AR.Drone model, differently from the common KF literature.

The duality between the LQR and the KF can be summarized by the following (Stevens et al., 2016):

$$\begin{array}{ccc} LQR & & KF \\ \mathbf{A_a} & \leftrightarrow & \mathbf{A_a}^T \\ \mathbf{B_a} & \leftrightarrow & \mathbf{C_a}^T \\ \mathbf{K} & \leftrightarrow & \mathbf{L}^T \end{array} \tag{18}$$

Then, in order to compute the Kalman optimal estimator gain $\mathbf{L}$, equations (13) and (14) can be used appropriately just by doing the substitutions as depicted in (18). The system to be observed, however, needs to be observable, which is the case for the augmented AR.Drone model.

By using the following weighting matrices, obtained by trial and error while evaluating the KF convergence to the unit-step and its Principal Gains in the frequency-domain,

$$\begin{aligned} Q_{kf} &= \mathbf{I}_{6\times 6} \\ R_{kf} &= diag \begin{pmatrix} 10^5 & 10^5 \end{pmatrix} \end{aligned} \tag{19}$$

the estimator gain $\mathbf{L}$ obtained is

$$\mathbf{L}^T = \begin{bmatrix} 0.0073 & 0.0031 & 0.0002 & -25.6\text{E}-6 & 0.1216 & 0.0300 \\ 0.0025 & 0.0033 & 0.0002 & -19.6\text{E}-6 & 0.0294 & 0.0784 \end{bmatrix} \tag{20}$$

4.3 LQG analysis

The Separation Principle is commonly applied to linear control systems and it allows the LQR system to be designed independently of the Kalman

filter and later to use them together as a dynamic compensator known as the LQG. The complete LQG system is simulated considering stochastic disturbances in the states and in the outputs. It is also considered that the AR.Drone model has a transmission delay, $d = 6$, or 6 periods of delay of $65ms$ each.

The model shown in (2) is simulated within a program loop in Matlab, along with the Kalman estimator system,

$$\begin{aligned}\hat{\mathbf{x}}_\mathbf{a}(k+1) &= (\mathbf{A_a} - \mathbf{LC_a})\hat{\mathbf{x}}_\mathbf{a}(k) + \mathbf{B_a}\Delta\mathbf{u}(k) + \mathbf{Ly}(k) \\ \hat{\mathbf{y}}_\mathbf{a}(k) &= \mathbf{C_a}\hat{\mathbf{x}}_\mathbf{a}(k)\end{aligned} \tag{21}$$

The estimated state vector of the augmented system is then passed to the LQR controller that computes the control increment law, given by

$$\Delta\mathbf{u}(k) = -\mathbf{K}\mathbf{x_a}(k) + \mathbf{K_y}\mathbf{y_r}(k) \tag{22}$$

Finally, the control increment is compensated using the inverse of the DC gain, $\mathbf{P}$, of the open-loop system, and sent to the input of the AR.Drone model equation as shown in (2):

$$\mathbf{u}(k) = \mathbf{u}(k-1) + \mathbf{P}\Delta\mathbf{u}(k) \tag{23}$$

The main simulation results are presented from figures 2 to 4. Unfortunately, due to the sake of compactness, the Principal Gains curves and robustness analysis will not be shown. However, it is possible to observe in figures 2 and 3 that the LQG designed is over-conservative, with a settling-time close to $16s$ and smooth control signals even under noisy output measurements. This conservative behavior has granted a successful and safe flight at the *first try* within the experimental environment to be described in the next section.

In Figure 4, only four estimated state variables are presented, which were considered more important to be covered in this analysis. Observe how the Roll and Pitch rates have been filtered and also how they are asymptotically recovered when a Roll Angle and a Pitch Angle are settled to maintain the speeds of the references. This is the Roll Rate and Pitch Rate CAS integrated in the Speed Hold design. This results are now going to be investigated with the real AR.Drone.

5 The experimental setup and results

The experimental setup is comprised of an AR.Drone linked to Mathworks' Matlab/Simulink using the *AR Drone Simulink Development-Kit V1.1* made by Sanabria and Mosterman (2014), addressed from now on as the DevKit.

The DevKit comes with simulation examples and real experimental diagrams for Simulink. The communication link between Simulink and the AR.Drone is completely independent from the

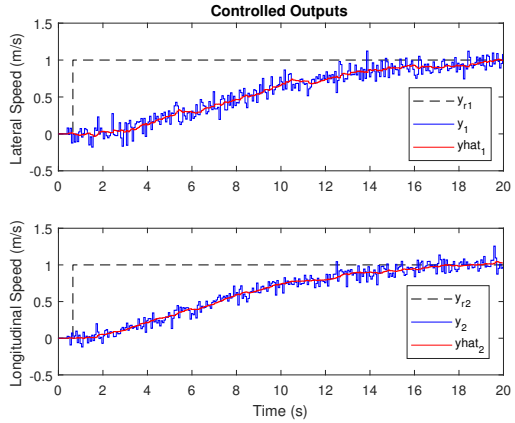

Figure 2: AR.Drone with LQG in simulation: controlled outputs.

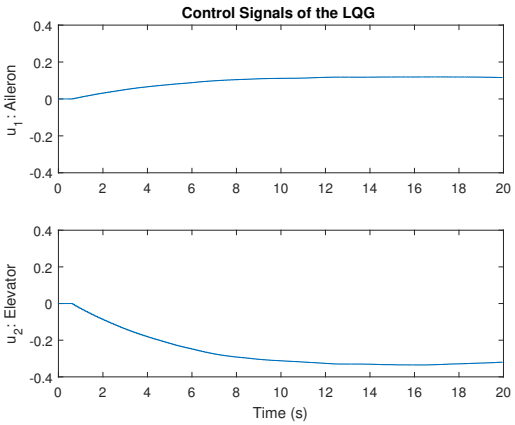

Figure 3: AR.Drone with LQG in simulation: control signals.

DevKit and is in fact a WiFi solution from Parrot, which allow the establishment of a TCP/IP connection between a client computer and the AR.Drone as the server. The DevKit works on sending and receiving data packets over this TCP/IP network and converting data types and organizing everything in a very simple and intuitive way for control systems engineers.

The LQG Speed Hold autopilot was embedded within a block called *Baseline Controller* in the DevKit, substituting all Proportional control systems within this block, except for the Proportionals Heading Hold and Altitude Hold autopilots of the DevKit's *Baseline Controller*.

The only experimental test presented in this work is the hovering flight or longitudinal and lateral speeds regulation. For the hovering test, in order to have a basis of comparison common to every AR.Drone user, the LQG system results are compared to the Hover WiFi example that comes with the DevKit.

The experimental results of the LQG and P controllers are shown in figures 5 and 6. Estimated states by the Kalman filter, available only for the

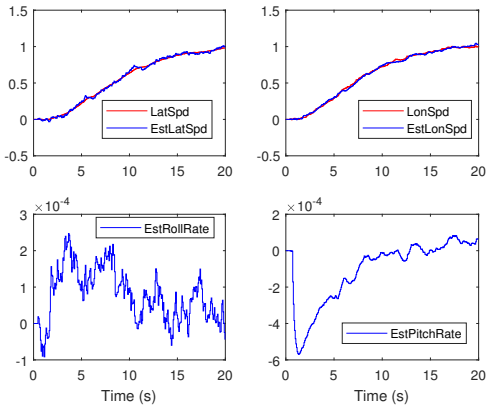

Figure 4: AR.Drone with LQG in simulation: estimated states.

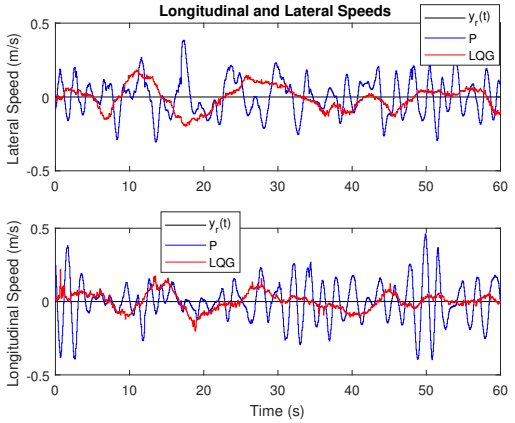

Figure 5: AR.Drone with LQG in real flight: controlled outputs.

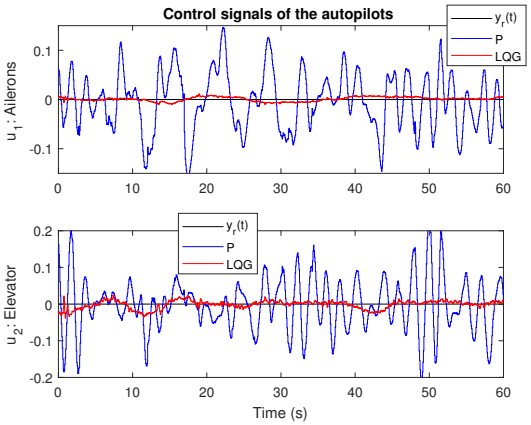

Figure 6: AR.Drone with LQG in real flight: control signals.

LQG results, are shown in Figure 7. Along with these results, in Table 1, some discrete quadratic indexes that quantifies how much power is within a data window, is used to verify the power of the lateral and longitudinal speed errors and the power of the control signals used to regulate these speeds. The ISE index is associated to the errors and ISU to the control signals, whilst both are computed based on the following incremental quadratic index:

$$J_{\text{power}} = (\mathbf{w}^{\mathbf{T}}\mathbf{w})T_s \tag{24}$$

where $\mathbf{w}$ is a dataset vector containing the information which the power is wanted to be known.

Observing the signals in figures 5 and 6 it is very clear that the LQG results could keep the speeds closer to zero despite the noisy measurements. The LQG is also more economic, since the control signals magnitudes are clearly smaller than the Proportional case. In terms of performance and efficiency/economy, these results are confirmed by the quantities shown in Table 1.

It is important, however, to remark that this comparison is between a MIMO LQG system and a two SISO P controllers and it is unfair indeed. However, this is just a preliminary work, at least with the AR.Drone, to present how a centralized MIMO LQG design could be done for this quadcopter, but this remark needs to be made, since several complex applications are appearing, day-by-day, with quadcopters among humans, and when humans are involved with aerial vehicles, search the history as shown by Stevens et al. (2016) and it is clear that at least for the AR.Drone there is plenty of space for tests with the standards of aerospace engineering.

6 Conclusions

This work has covered the design and application of a Speed Hold autopilot for the Lateral and Longitudinal dynamics of an AR.Drone 2.0 quadcopter system. This autopilot had integrated Roll Rate and Pitch Rate control augmentation systems and the design has also covered how to model the airframe based on registered flight data and the application of the Recursive Least-Squares in state-space. The control design method adopted is a standard in the flight control systems literature, the Linear Quadratic Gaussian method.

The major advantage of MIMO LQG is the time required to design and implement complex multivariable control systems, which is a short time, when the design technique and the use of linear algebra is an everyday tool for the control

Table 1: Performance indexes for Lateral and Longitudinal dynamics.

Ctrl. Type	P	LQG
LatSpd ISE	0.9364	0.39228
LatCmd ISU	0.21532	0.0011896
LonSpd ISE	1.1502	0.20757
LonCmd ISU	0.3212	0.0076623

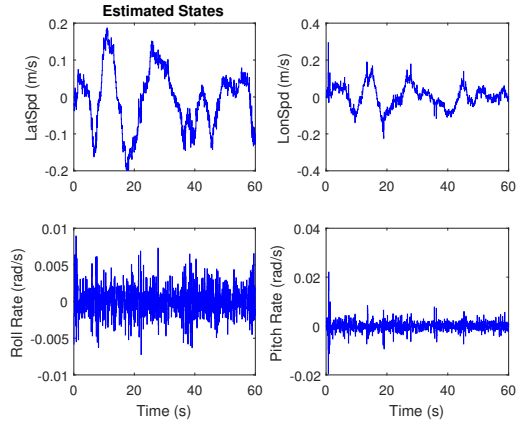

Figure 7: AR.Drone with LQG in real flight: estimated states.

engineer. This advantage comes from the solution of an optimization problem, to all state variables at once, by solving a single algebraic equation in the LQR and Kalman filter design.

The drawback of the centralized MIMO LQG design is its mathematical background complexity and the slow learning curve of such techniques involved. These are generally passed on to engineering students during the Graduate stage only and the availability of a real process with 6-DOF to study and apply the learned concepts of linear systems theory is very rare. In general, those who study the MIMO LQG deeply are somehow doing research on GNC problems for aerospace systems, while quadcopter drones are now being studied and used by a much wider group in every niche of science and final applications will be compromised if badly designed flight control systems are being used at the stability level of these drones.

This study with the AR.Drone will continue in several directions. The next step is a complete MIMO LQG system encompassing the Altitude Hold and Heading Hold autopilots, substituting the whole *Baseline Controller* block from the DevKit. It is also underway the development of a local positioning system based on low cost depth sensors that would allow Guidance experiences to be conducted for waypoint tracking. It is also expected that in the following years model following design techniques will be implemented and embedded within the AR.Drone firmware using a newer development kit for Simulink. This kind of tests will allow experiences of dynamic emulation of microgravity systems within the lab, or how would be to pilot a quadcopter within different pressurized scenarios, among many other control systems experiences that might appear during this journey.

Acknowledgments

The authors gratefully acknowledge the financial support of the Brazilian Research Agency, the CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*), registered under the process number 408559/2016-0.

References

Armah, S. and Yi, S. (2015). Altitude regulation of quadrotor types of uavs considering communication delays, *IFAC-PapersOnLine* **48**(12): 263 – 268.

Bitmead, R. R., Gevers, M. and Wertz, V. (1990). *Adaptive optimal control: the thinking man's GPC*, Prentice Hall.

Clark, R. A., Punzo, G., MacLeod, C. N., Dobie, G., Summan, R., Bolton, G., Pierce, S. G. and Macdonald, M. (2017). Autonomous and scalable control for remote inspection with multiple aerial vehicles, *Robotics and Autonomous Systems* **87**: 258 – 268.

Criado, R. M. and Rubio, F. R. (2015). Autonomous path tracking control design for a comercial quadcopter, *IFAC-PapersOnLine* **48**(9): 73 – 78.

Hernandez, A., Murcia, H., Copot, C. and DeKeyser, R. (2014). Model predictive path-following control of an ar.drone quadrotor, *XVI Congreso Latinoamericano de Control Automático, CLCA 2014*, Vol. 1.

Mac, T. T., Copot, C., Keyser, R. D. and Ionescu, C. M. (2018). The development of an autonomous navigation system with optimal control of an uav in partly unknown indoor environment, *Mechatronics* **49**: 187 – 196.

Malik, M. B. (2004). State-space recursive least-squares: Part i, *Signal Processing* **84**(9): 1709 – 1718.

Sanabria, D. E. and Mosterman, P. J. (2014). ARDrone Simulink Development Kit v1.13, mathworks.com/matlabcentral/fileexchange. Accessed: March 31, 2018.

Santos, M. C. P., Santana, L. V., Brandão, A. S. and Sarcinelli-Filho, M. (2014). Controle e estimação de posições 3d de um vant com um sistema de captura usando uma câmera de profundidade, *IEEE/IAS INDUSCON 2014, Juiz de Fora - MG.*

Santos, M. C., Santana, L. V., Brandão, A. S., Sarcinelli-Filho, M. and Carelli, R. (2017). Indoor low-cost localization system for controlling aerial robots, *Control Engineering Practice* **61**: 93 – 111.

Stevens, B. L., Lewis, F. L. and Johnson, E. N. (2016). *Aircraft Control and Simulation: dynamics, controls design, and autonomous systems*, 3rd edn, John Wiley & Sons, Inc.