Identification and Path Following Control of an AR.Drone Quadrotor

Andres Hernandez, Cosmin Copot and Robin De Keyser
Department of Electrical energy
Systems and Automation
Ghent University, Belgium
Email: Andres.Hernandez@UGent.be

Tudor Vlas and Ioan Nascu
Department of Automation
Technical University of Cluj-Napoca
Romania
Email: tudorv90@yahoo.com

***Abstract*—This paper describes the process of identification and closed-loop control of an Parrot AR.Drone Unmanned Aerial Vehicle (UAV) as well as a path following application based on IMC position controllers. The research issue is to achieve position control of the AR.Drone quadrotor movement via its on-board sensory equipment and external webcam video stream. Firstly, transfer functions are detailed for pitch and altitude movements and a comparison is made between implemented PID and IMC controller performance for both simulation and practice. Furthermore, using IMC controllers, a path following application exhibits controller behavior from a practical point of view. It is concluded that the dynamic model and the controllers implemented on the quadrotor can serve as a reliable basis for more advanced applications.**

I. Introduction

In recent years there has been increased interest in autonomous aerial vehicles that can execute complex tasks without sustained human supervision. To undertake the challenging task of automated flight and maneuvering, the design of a versatile flight control is required. One type of aerial vehicle which can accomplish this is a quadrotor. The quadrotor is a micro Unmanned Aerial Vehicle (UAV) which comes equipped with on-board sensory equipment and communicates wirelessly with a command station. It has four rotating blades which enable flight in a way similar to that of a helicopter. Movement is attained by varying the speeds of each blade thereby creating different thrust forces. The quadrotor can fly both indoor and outdoor and is able to perform aggressive aerial maneuvers as well as keep stable and precise flight paths. In this study, we use an AR.Drone 2.0 model from the French company Parrot®. This model is available to the mass market and was chosen thanks to its simple structure, sufficient sensory equipment and ease of maintenance, at a very low price. These features prove the quadrotor to be a good subject for both study purposes and practical applications.

A large number of papers emerged in literature on this topic in the past years. Modeling, Identification and Control of a quadrotor for autonomous control is described by [1] with use of on-board sensing. Also, Simultaneous Localization and Mapping (SLAM) was implemented to create 3D maps of the environment as well as to establish quadrotor position in space [2]. A wide area of papers on the quadrotor implement various visual methods for automatic navigation and object recognition with filtered data from on-board sensors and cameras [3]. In more advanced studies there are used external camera systems, more accurate sensory equipment, and a more advanced type of quadrotor (AscTec Pelican) to estimate the trajectory of an object and catch it [4]. In the previously mentioned case, Model Predictive Control control was studied and successfully used.

978-1-4799-2228-4/13/$31.00 ©

The contribution of this paper is to give the transfer function, control parameters and structure for quadrotor automated flight adding proof of their validity by an experimental approach. The main sections exhibit the dynamic model of the AR.Drone 2.0 quadrotor, identification and validation of the model, design of suitable PID and IMC controllers. The identification and control sections mentioned deal with pitch and altitude dynamic model and both PID and IMC control for both movements.

To make clear what the end objective is, we must describe the notion of path following [5]. Path following requires the quadrotor to follow a desired geometric path, implying a constraint in space, but not in time. Thus, the time it takes the quadrotor to reach the target position does not matter here. Path following is implemented by use of position controllers in 2D vertical plane. The combined on-board sensors and a web-cam are used to follow a reference geometric path in the visual frame.

II. AR.Drone 2.0 Quadrotor Platform

The AR.Drone 2.0 is a low-cost micro Unmanned Aerial Vehicle. Its producer offers a detailed Developers Manual [6] along with a Software Development Kit. There is also much information and insight to be found on the internet on AR.Drone amateur forums and websites. The quadrotor comes with internal in-flight controllers and emergency features making it stable and safe to fly [7]. Thus, there are sufficient arguments to buy such a device instead of building one with the only downside being that access to the internal controllers is restricted. The internal software is black-box and the parameters that refer to control, motors and other calibrations are undocumented.

A. Hardware Frame

The AR.Drone 2.0 has a classical quadrotor design, with four propeller blades arranged symmetrically around a central unit which holds the sensory equipment and the circuit board. The parts are held together by a carbon-fiber frame and highly

resistant PA66 plastic. For protection of the propellers in indoor flight, a detachable foam hull is used, giving a total weight of 420g. For outdoor flights, a different, lighter hull is used weighing the quadrotor at 380g [6].

There are 4 brushless DC motors powered by 14.5 W each from the 3 element 1000 mA/H LiPo rechargeable battery which gives an approximate flight autonomy of 10-15 minutes. Two video cameras are mounted on the central hull. The front camera resolution is 1280x720 and the bottom one is 640x360 with a video stream rate of 30 FPS and 60 FPS for front and bottom cameras.

B. Electronics and Communication

There are 2 main circuit boards: The Mother-board holds the ARM9-core, 32 bit, 468MHz processor which runs with 128 MB DDRAM at 200MHz frequency on a Linux-based real-time operating system. The processor acquires data flow from the video cameras and it also has a USB connector for add-ons. The second board is a 16-bit micro-controller Navigation board which interfaces with the sensors at a frequency of 40 Hz.

The sensors are located below the central hull and consist of a 3-axis accelerometer, a 2-axis and a 1-axis gyroscope which together form the Inertial Measurement Unit (IMU). There are two ultrasonic sensors and a pressure sensor for altitude estimation. Communication between quadrotor and command station is done by Wi-Fi connection within a 50 m range via three separate communication channels. For more details about internal structure, check [6] [7].

C. The command station

The AR.Drone 2.0 is designed to be controlled directly by smartphone or tablet. In the case of this study, the quadrotor is controlled from a computer by a Visual Studio C++ program. OpenCV, FFmpeg and Pthreads libraries are used for image and video processing and thread handling. The program at the basis of development of quadrotor control and data manipulation is open-source and was developed by an AR.Drone enthusiast and posted on a topic-related forum. For source code see [8]. The application establishes access to all AR.Drone communication channels, enabling functions to send commands or set configurations, receive and store data from sensors and video stream. Thus, data can be interpreted off-line for the purpose of identification, modeling and control of the quadrotor.

III. Quadrotor Dynamics and Identification

The quadrotor aerial movements are similar to those of a conventional helicopter. The difference is that movement is achieved by varying each of the four motor speeds to obtain a desired effect that causes movement. Fig. 1 depicts the movement axes of the quadrotor with its six degrees of freedom (DOF). Quadrotor movements are explained as following:

- Pitch - Rotational movement along transversal axis y generating a translational movement on x axis.
- Roll - Rotational movement along longitudinal axis x generating a translational movement on y axis.
- Yaw - Rotational movement along z axis. The quadrotor rotates about its vertical axis.
- Throttle - Translational movement on z axis. The quadrotor ascends or descends.

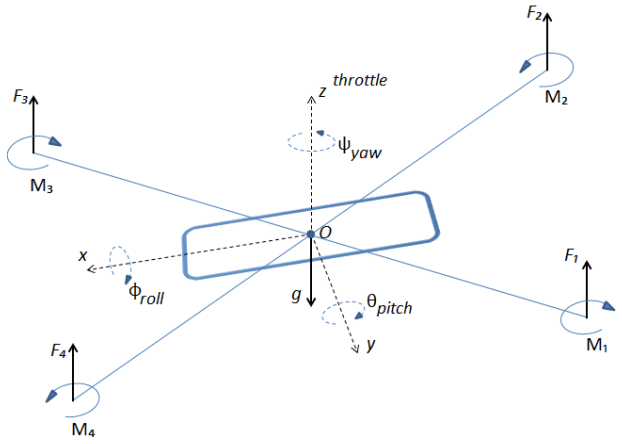

Fig. 1. Quadrotor Movement Axes

Thus, the quadrotor can fly to any given point in space either by a sequence of different movements or by combining several movements together. Movement is achieved by giving reference values as input to the internal, black-box controllers [6]. The input and output relations will be discussed in the following subsection.

A. Quadrotor movement, inputs and outputs

As mentioned previously, access to the motor supply or the internal controllers is restricted and their structure is not fully documented. For more details about sensor calibration and embedded functions check [6].

The reference parameters given to the internal controllers are scaled floating point values between [-1, 1] representing the percentage of the minimum or maximum configured angle or speed for the respective movement. We denote inputs by: $\{\phi_{in}, \theta_{in}, \dot{\zeta}_{in}, \dot{\psi}_{in}\}$ - the roll angle reference value, pitch angle reference, vertical speed reference and yaw angular speed reference. The output values measured by the quadrotor sensory equipment are denoted by: $\{\phi_{out}, \theta_{out}, \psi_{out}, \zeta_{out}, \dot{x}, \dot{y}\}$ symbolizing roll, pitch and yaw angle in radians, altitude in meters and linear velocities on longitudinal and transversal axes in m/s.

Fig. 2. Inputs and Outputs of Quadrotor

On-board controllers mostly deal with decoupling of the system so that each movement can be identified separately

as a single-input single-output (SISO) system. As observed experimentally, coupling effects occur mostly at limit functioning values of quadrotor movements i.e. high speeds or very complex maneuvers.

Data received from the video stream is combined with sensor estimations to compute the quadrotor's approximate position and speed. Although drift in the sensors exists, it has been proven in literature that using an Extended Kalman filter to fuse data from both video stream and sensors increases the accuracy of the estimated position and speed. For more details see [7], [9].

B. Parametric Identification - Pitch

The quadrotor is treated as a Linear Time-Invariant System and identification is done over pitch, throttle and yaw movements. Due to the symmetry of the quadrotor, roll movement shares the same model with pitch therefore identification is not done for it separately. A main sampling loop of $30ms$ is implemented for all input/output operations.

In the case of pitch movement, the parametric identification is done around a null operating point (middle of range [-1, 1]). From the step response of the system, the pitch time constant is determined to be $\tau_c = 1.8s$ yielding a sampling time:

$$T_s = \frac{\tau_c}{10} \approx 180ms \tag{1}$$

. The time delay $T_d = 360ms$ is quite large and may come from the reaction time of the internal controller to modify pitch angle and achieve movement. A plot of the step response is shown in Fig. 3.

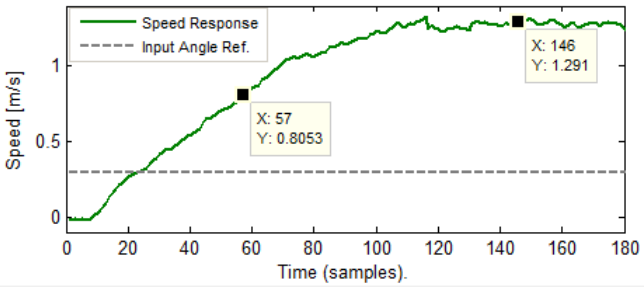

Fig. 3. For an input signal of pitch angle factor between [-1, 1] of 0.3, the measured output speed is plotted. The time constant value is computed at 63.2% of Steady state speed value of $1.29[m/s]$

The transfer function gives the relation between input pitch angle reference [-1, 1] and output measured linear speed on pitch. The choice of the identification input signal is a Pseudo-Random Binary Signal (PRBS) generated in Matlab. The PRBS is applied for a period of 200 samples with $T_s = 180ms$. Data from several tests are saved and used for estimation and validation of the transfer function. To estimate the transfer function, Prediction Error Method (PEM) [10] is used in Matlab. The obtained pitch angle to speed transfer function in discrete form is:

$$H_{spd}(z) = \frac{0.356}{1 - 0.9079z^{-1}}(z^{-3}) \tag{2}$$

The estimated transfer function is of first order and gives a fitting of 55.34%. The minimum and maximum amplitudes of the input PRBS are $\{-0.3, 0.3\}$.

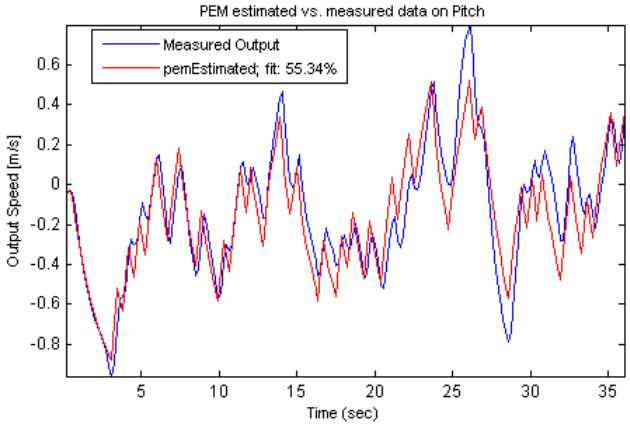

Fig. 4. The comparison of the PEM-estimated transfer function response and measured data (speed in m/s) is shown in Fig. 4 with a fitting of 55.34%

The transfer function approximates well the real plant behavior as seen in Fig. 4. The transfer function can be used for designing a speed controller directly or a position controller by adding an integrator in its structure. The position controller is detailed in Section V. In the next subsection, a similar process of finding the transfer function for altitude movement is applied.

C. Parametric Identification - Altitude

In the case of altitude, we have a system with an integrator, since the input is a speed reference while the output is position (sensors do not provide throttle speed estimate). To determine the time constant, a step is applied, and the altitude measurements are differentiated to obtain speed. The resulting step response in speed is $\tau_c = 0.95s$ and a time delay of $T_d = 0.27s$. The chosen sampling time is:

$$T_s = \frac{\tau_c}{10} \approx 90ms \tag{3}$$

A Swept-Frequency Cosine signal (Chirp) is designed with frequency range of $0.1 \rightarrow 1Hz$ for a period of 300 $samples$. The Prediction Error Method [10] estimated function has the discrete form:

$$H_{alti}(z) = \frac{0.01257 + 0.01108z^{-1}}{1 - 1.684z^{-1} + 0.684z^{-2}}(z^{-5}) \tag{4}$$

Validation is done the same as in the case of pitch. The estimated transfer function exhibits a fitting of 68.8% (see Fig. 5) and cross validating with other sets of data yields fittings close to the original.

IV. Control Methodologies

In this paper, control of the quadrotor is described for pitch and altitude position. Two types of controllers are designed based on the identified transfer functions. In the case of pitch, the transfer function has an integrator added. We describe the methodologies for PID and IMC control.

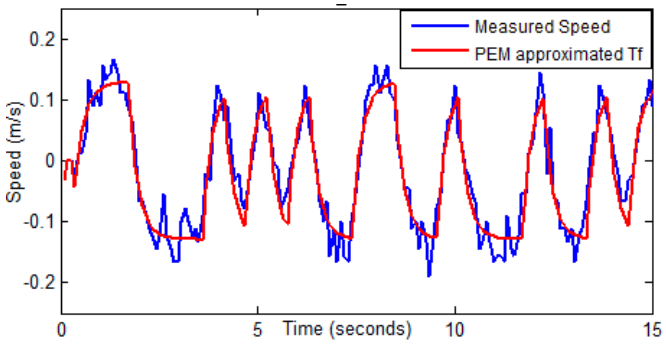

Fig. 5. PEM estimated Transfer Function fitting on measured speed. The measured speed is obtained by differentiating the altitude response for an input chirp signal

A. PID design using CAD tools

As the PID definition and structure are well known, it will not be detailed in this paper. We mention however the use of anti-reset windup and placing the derivative component on feedback to avoid derivative kick as described in [11].

The closed loop structure of the system is a conventional feedback structure. Since the process model is available, computer aided design (CAD) tools are used to design the controller. The Frequency Response toolbox (FRtool) in Matlab is used as described in [12]. With this graphical interface, controller design specifications are set and the controller parameters (in this case, a PID) are computed.

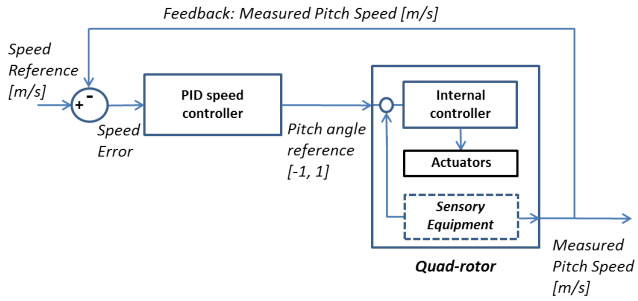

Fig. 6. Scheme of the Pitch closed loop model with speed control. The structure is the same for the other controllers but with changed controller parameters.

B. Discrete-time IMC work principles

The IMC is a model-based controller which relies on the use of the internal plant model [13] [14]. The IMC controller performance is dependent on how well the model approximates the actual process.

An IMC closed loop control scheme is depicted in Fig. 7, where $\frac{B(q^{-1})}{A(q^{-1})}$ is the system PTF (Pulse Transfer Function), the discrete-time equivalent of the process transfer function. $F(q^{-1})$ is a filter which ensures a proper compensator. In the IMC context, the controller is designed based on compensating the process dynamics while ensuring desired closed loop performance. This leads to the following design algorithm of the IMC controller:

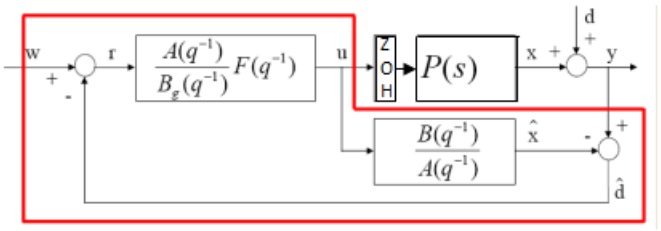

Fig. 7. Schematic overview of the IMC closed loop control scheme.

First, split the process transfer function $\frac{B(q^{-1})}{A(q^{-1})}$ in an invertible (good) part and a non-invertible (bad) part. This implies that if the transfer function has time delays or zeros outside the unit circle of z-plane, they are thus part of the non-invertible (bad) part of the process, denoted as $B_b(q^{-1})$; with the normalization $B_b(1) = 1$. Second, the remaining invertible (good) part of the process is then denoted by $\frac{B_g(q^{-1})}{A(q^{-1})}$.

A 'basic' IMC filter, designed to follow step changes in the setpoint and to reject step disturbances at the output of the process, is given by:

$$F(q^{-1}) = \frac{(1+a)^n}{(1+a\cdot q^{-1})^n} \tag{5}$$

with steady state gain $F(1) = 1$ and a a design parameter defined as

$$a = -e^{-T_s/\lambda} \tag{6}$$

The values of this design parameter are in the range $0 << |a| < 1$ and it is related to the closed-loop speed: if λ is big, the $|a|$ is closer to 1, making the controller less aggressive, with a higher settling time. The case of the extended filter is not described in this paper.

We can now describe the IMC controller as an equivalent $R(q^{-1})$ with structure given by:

$$R(q^{-1}) = \frac{A(q^{-1})F(q^{-1})}{B_g(q^{-1}) - B(q^{-1})F(q^{-1})} \tag{7}$$

The discrete-time IMC controller $R(q^{-1})$ can be represented into an equivalent closed loop control scheme as depicted in Fig. 8.

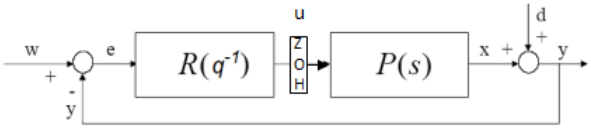

Fig. 8. Equivalent Discrete-IMC representation for $R(q^{-1})$.

V. AR.DRONE CONTROL - PITCH

Having the dynamic model of the quadrotor, control can now be achieved for different end objectives. Each controller can be implemented for either position or speed references. Firstly, PID control is designed, tested in simulation and on the real plant. Internal Model Control (IMC) is also implemented and tested in order to compare performance with PID. We will start by describing the design process and implementation of

the PID and finish with IMC and a comparison between the two for both pitch speed and altitude position control.

A. PID - Pitch

A brief description of the PID design for Pitch Position control is given. Following the steps described in IV-A, the controller is designed based on the discrete time transfer function:

$$H_{pos}(z) = \frac{0.03623z^{-1} + 0.03478z^{-2}}{1 - 1.884z^{-1} + 0.8844z^{-2}}(z^{-5}) \quad (8)$$

The PID parameters found with FRTool are: $K_p = 0.2333, K_i = 0.04559, K_d = 0.2985$ and the sampling period is $T_s = 180ms$. The PID is tested in simulation and in real case for a reference of $3m$ with the results plotted in Fig. 9.

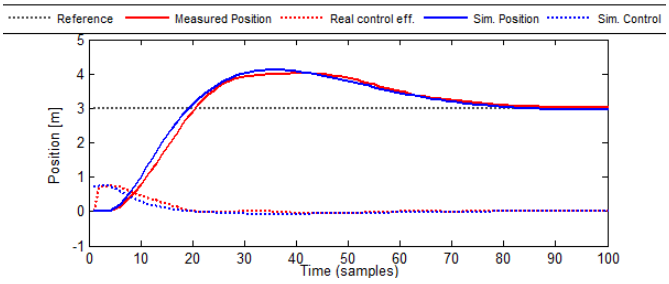

Fig. 9. Comparison of Simulation and Real PID step response

It can be observed that measured position is very close to the simulated one. The control efforts are also very similar. However, there is undesired overshoot $OS > 30\%$ and $T_{set} > 14s$. It is clear that performance must be improved therefore an IMC is designed and implemented in the next subsection.

B. IMC - Pitch

Following the algorithm described in subsection IV-B, an IMC controller is tuned for pitch movement. A strong advantage of the IMC controller is that the tuning parameter a can be changed and the $R(q^{-1})$ recomputed for different types of requirements. In the case of following a fixed reference or small step changes (i.e. following fixed trajectory, stable slow maneuvers) a basic-filter with a close to -1 will give a slower but more stable control.

As the non-invertible bad part consists of the time delay, it remains that the invertible part is used to calculate the controller of form $R(q^{-1})$. The IMC parameters are $a = -0.85$:

$$R(q^{-1}) = \frac{0.3169q^{-1} - 0.2802q^{-2}}{1 - 0.7q^{-1} + 0.0225q^{-2} + 0.0225q^{-3} + 0.011q^{-4}} \quad (9)$$

To evaluate how well the IMC performs, we set the same $3m$ reference for both IMC and PID controlled systems and compare them.

As it can be seen in Fig. 10, the IMC eliminates overshoot and provides a much faster settling time of $T_{set} < 9s$ as compared to the PID $T_{set} \approx 14s$.

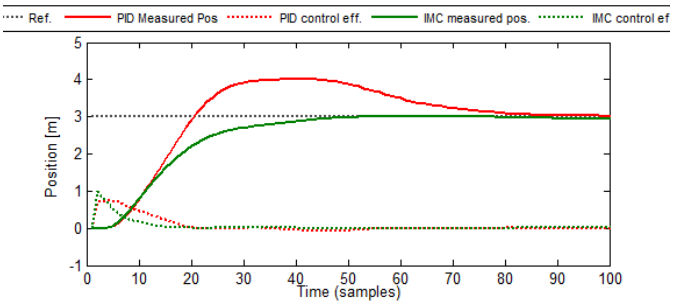

Fig. 10. Pitch PID vs. IMC comparison

VI. AR.DRONE Control - Altitude, Final Application

We will also exemplify the altitude position control. Using the transfer function identified in subsection III-C, PID and IMC controllers are designed and compared in the case of altitude position control.

A. PID - Altitude

The PID is designed following the same steps as in the case of pitch. However, the altitude controller should have minimum overshoot as well as a fast settling time. Because the bottom camera acts as a sensor, holding an altitude reference stable is paramount. The PID parameters found with FRTool are: $K_p = 1.822, K_i = 0.97, K_d = 0.8513$ and the sampling period is $T_s = 90ms$. In Fig. 11, a set point of $1.2m$ is given. As it can be observed, there is a large overshoot $Os > 30\%$ and $T_{set} > 10s$.

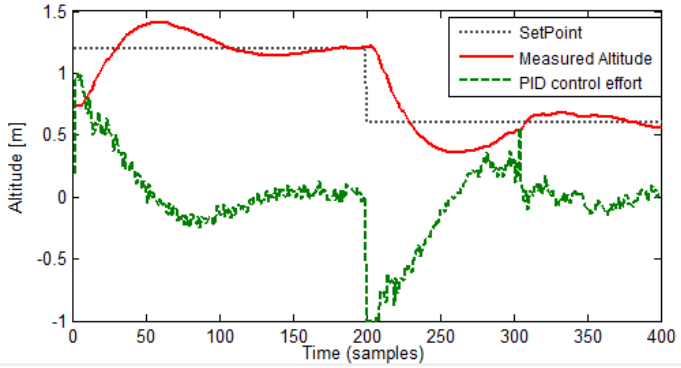

Fig. 11. Altitude PID controlled position

In both the cases of Altitude and Pitch, the PID controller cannot ensure satisfactory specifications because the large overshoot is undesired in an indoor environment.

B. IMC - Altitude

In the case of altitude, the bad part is again represented only by the time delay while the invertible part of the discrete time transfer function sampled at $T_s = 90ms$ is :

$$\frac{B_g(q^{-1})}{A(q^{-1})} = \frac{0.01257q^{-1} + 0.01108q^{-2}}{1 - 1.684q^{-1} + 0.684q^{-2}} \quad (10)$$

The same method of discrete-IMC design is used as described in the case of pitch. Here is an example of an IMC controller with tuning parameter $a = -0.6$:

$$R(q^{-1}) = \frac{6.7638q^{-1} - 4.6265q^{-2}}{1 - 0.2q^{-1} + 0.16q^{-2} + 0.16q^{-3} + 0.0749q^{-4}} \quad (11)$$

In Fig. 12, we compare the IMC and PID performance:

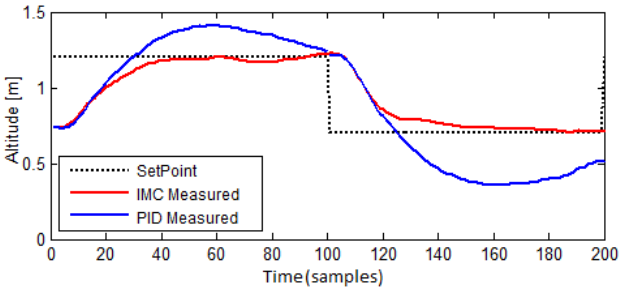

Fig. 12. Altitude PID vs. IMC comparison for set points of $1.2m$ and $0.6m$ given for a period of 100 samples (9s) each.

There is a visible performance increase from PID to IMC. Settling time is reduced to $T_{set} \approx 4.5s$ from $10 - 12sec$ and $OS = 0$ or very small compared to $OS \approx 30 - 50\%$ with the PID.

Overall, the usage of IMC employs an easier method of changing control performance with respect to specifications, due to the availability of the design parameter a. Adding also the performance improvements over the PID, we conclude that the IMC offers a better control for the system.

C. Path Following application

The quadrotor can follow a fixed path in space in several ways. A simple yet fairly efficient method is the usage of an external video device which provides the position of the quadrotor in 2D or 3D space. By using a basic color segmentation algorithm to recognize the quadrotor, one or several cameras can accurately provide its position relative to the camera frame. Once the quadrotor position is known, coordinates in the frame of the webcam can be given as set points to the drone controllers. In our case, a low-cost webcam was used and a two-dimensional trajectory given to the quadrotor.

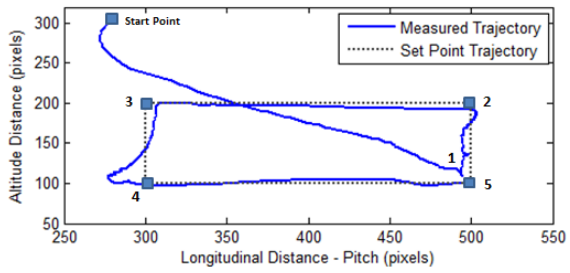

Fig. 13. Quadrotor executes a square trajectory in 2D Vertical Plane

In Fig. 13, a square path is shown in terms of altitude and pitch position. The quadrotor travels from Start Point to $point5$ via intermediate checkpoints. Note that the trajectory is in vertical 2D plane. Basic-filter position IMC is used for Pitch and Altitude movements to follow a position reference provided by webcam.

VII. Conclusion

This paper offers a point of start for more advanced applications involving automated control of an AR.Drone quadrotor. The dynamic model of the quadrotor is found by means of parametric identification and control designed for the case of position references. A comparison between PID and IMC is made for both pitch and altitude showing an increase in performance with IMC use. A basic application implemented in practice demonstrates control behavior for image based path following.

As future work, a more advanced type of controller - Model Predictive Control (MPC) - may be implemented over a basic speed control structure in order to achieve trajectory tracking, implying both space and time constraints. Also, a refinement of the path tracking application can be made to obtain more precise control by filtering sensory readings (if on-board sensors are used) or implementing a more advanced external video tracking system.

References

[1] Y. Sun, (2012). Modeling, Identification and Control of a Quad-rotor drone using low-resolution sensing.*Master of Science in Mechanical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign*

[2] N. Dijkshoorn, (2012). Simultaneous localization and mapping with the AR.Drone. *Masters Thesis for the graduation in Artificial Intelligence, Universiteit van Amsterdam*

[3] M. Mogenson, (2012). The AR Drone LabVIEW Toolkit: A Software Framework for the Control of Low-Cost Quadrotor Aerial Robots, *Master's Thesis in Mechanical Engineering, Tufts University*

[4] P. Bouffard, (2012). On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments, *Technical Report No. UCB/EECS-2012-241*

[5] E. Xargay, V Dobrokhodov, I. Kaminer, et al. Time Critical Cooperative Control of Multiple Autonomous Vehicles (2012), *IEEE Control Systems Magazine, October 2012*

[6] S. Piskorski, N. Brulez, P. Eline, F. D'Haeyer, (2012). AR.Drone Developer Guide, *SDK 2.0*

[7] P.-J. Bristeau, F. Callou, D. Vissiere, N. Petit, (2011). The Navigation and Control technology inside the AR.Drone micro UAV *Preprints of the 18th IFAC World Congress Milano August 28 - September 2, 2011, pp. 1477-1484*

[8] N. Shimpuku (puku0x), (2013). CV Drone *Free Software* on https://github.com/puku0x/cvdrone

[9] J. J. Engel, (2011). Autonomous Camera-Based Navigation of a Quadrocopter. *Masters Thesis in Informatik, Technischen Universitat Munchen*

[10] L. Ljung. , (1999). System Identification: Theory for the user, *NJ: Prentice Hall Information and System Sciences Series*

[11] K. J. Åström, Tore Håglund, (2006). Advanced PID Control, *ISA-The Instrumentalion, Systems, and Automation Society*

[12] R. De Keyser, C.M. Ionescu, FRTool: a frequency response tool for CACSD in Matlab, in Proc. of the IEEE Conf. on Computer Aided Control Systems Design, Munich, Germany, 2006, 2275-2280.

[13] Bequette, B.W., (2003). Process Control: Modelling, Design and Simulation, *Prentice-Hall, Upper Saddle River*, NJ.

[14] M. Morari, E. Zafirou , (1989). Robust Process Control, *Englewood Cliffs (N.J.) : Prentice-Hall, 1989.*