Open-source Multi-UAV Simulator for the ROS Environment

Ignacio Mas*†
Sebastián Curi*
Ricardo Sánchez Peña*†

*Centro de Sistemas y Control
Instituto Tecnológico de Buenos Aires (ITBA)

†Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Email: iamas@itba.edu.ar

***Abstract*—This article presents an open-source simulator for multiple Ar.Drone quadrotors that runs under the ROS operating system. The simulator includes dynamic models of the commercially available Parrot Ar.Drone vehicles, and models of their onboard sensors with the addition of a simulated GPS receiver. The system was verified by the implementation of a 3-UAV formation control architecture and results for a step response and a trajectory following test are shown. The interface of the simulator with the ROS environment is similar to that of the open-source driver typically used to connect through ROS with the real vehicles. This feature allows for a seamless transition between simulation, *hardware-in-the-loop* testing and full hardware experimentation.**

I. Introduction

Advances in different fields of technology opened a wide range of possibilities in the utilization of unmanned aerial vehicles (UAVs) for activities such as remote monitoring, border patrol and surveillance, disaster relief, forest fire detection or archeology.

In general, these applications are preformed by a single vehicle that carries the onboard instruments necessary to execute the mission.

In recent years, there is a growing interest in replacing a single complex vehicle with a platoon of smaller, simpler and cheaper UAVs that together can achieve the same goals or even extend the overall capabilities when operating in a cooperative fashion.

In this article, we present an open-source simulator for multiple commercially-available Parrot Ar.Drone quadrotors. The simulator runs under the ROS operating system. The simulator includes dynamic models of the Ar.Drone vehicles and models of their onboard sensors, together with a simulated GPS receiver. The system is verified by controlling a platoon of three Ar.Drone quadrotors.

II. Background

The context of this work is the development of an autonomous aerial platform that can support research activities, as well as the demonstrations of applications and proof-of-concepts that can be of interest to the industry in areas ranging from agriculture and oil and gas exploration to remote sensing or motion picture film-making.

In particular, the overall objective of this work is the creation of an integral multi-UAV testbed, consisting of a simulation environment and a hardware system, that can provide a simple and fast development platform to study guidance, navigation and control of platoon of autonomous aerial vehicles, with emphasis on the formation control methods, the coordinated navigation and the early demonstration of potential applications.

Many institutions around the world have created their own testbeds to support multi-UAV research. Among them we can cite those that use quadrotors in indoor environments , such as the University of Pennsylvania [1] or MIT [2], or outdoors using helicopters, like the one developed by UC Berkeley [3]. Brigham Young University and Stanford University use fixed-wind UAVs [4], [5]. The testbed platform developed by ETH de Zurich allows for acrobatic flights in a protected enclosed environment [6].

In parallel with these developments, in the last few years a new software paradigm for robot applications has emerged, taking a strong position in the academic world: the ROS environment. ROS (Robot Operating System) is an open-source meta-operating system for robots that provides a communication layer and a data and programming structure that simplifies the development, the collaboration and the exchange of research tools and algorithms [7]. This system is being widely adopted by the scientific community, which allows for an unprecedented direct access to state-of-the-art developments in the robotics field.

This project makes use of the ROS operating system to integrate in a simple way different vehicles, personal computers and programming languages –such as C, MATLAB and Python–, and reuse community-developed robot drivers and tools. Additionally, the seamless integration of ROS and the Gazebo Simulation environment allows for high fidelity simulation and *hardware-in-the-loop* testing.

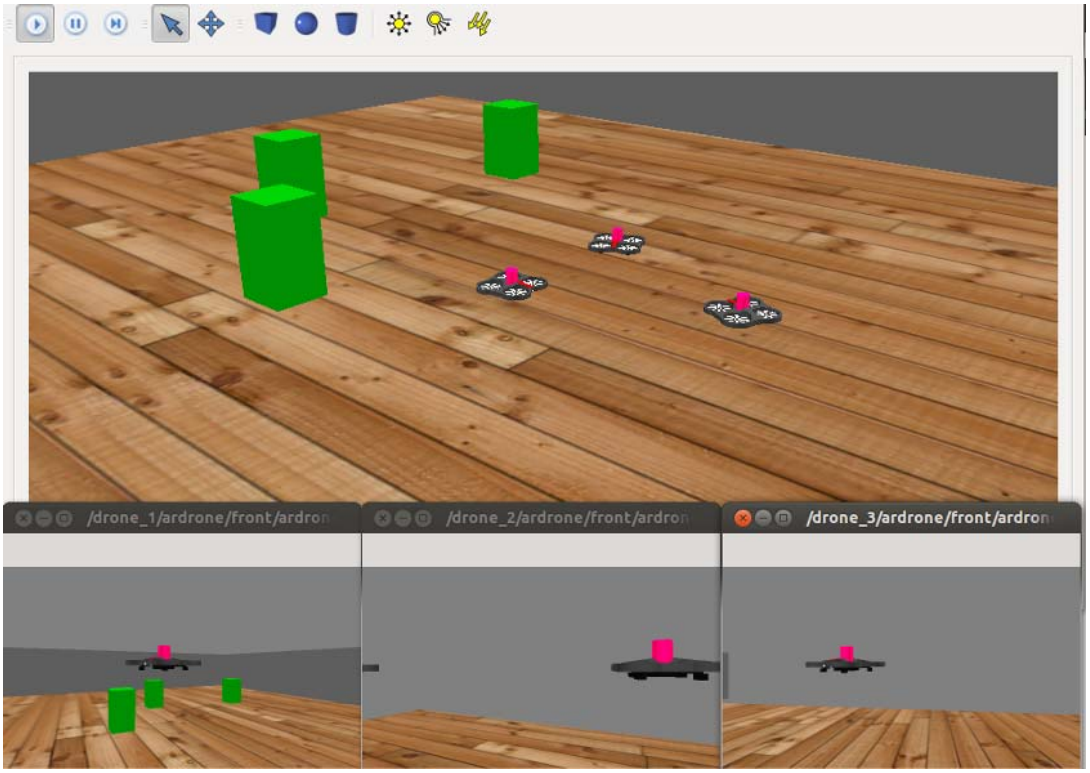

Fig. 1. Front-end of the developed multi Ar.Drone quadrotor simulator with front-camera capability for position estimation. The bottom windows show the on-board video image generated by each quadrotor.

III. Multi-UAV Simulation Environment

A high-fidelity Ar.Drone quad-rotor simulator with multi-vehicle capability was developed based on the open source single-quadrotor ROS package created under the Gazebo simulation environment by the Computer Vision Group at the Technical University of Munich (TUM). The original package was extended to add a functionality that was not previously available.

The simulator includes plugins that describe the different sensors onboard the commercially-available quadrotor: an inertial navigation system with 3-axes accelerometers and 3-axes gyroscopes, a front-view wide-angle camera and a low resolution bottom-view camera [8]. Additionally, a GPS receiver was simulated for each vehicle to provide a capability that is not part of the commercial system, but that we are in the process of integrating to our quadrotors to have absolute position measurements. The interface between the Gazebo simulation and the ROS environment is similar to that provided by the Ar.Drone ROS driver called `ardrone_autonomy`, which is based on the official AR-Drone SDK version 2.0 and supports both AR-Drone 1.0 and 2.0. This is the driver we currently use in the lab to operate the Ar.Drone quadrotors. This feature allows for a seamless transition between simulation and hardware experimentation. The driver `ardrone_autonomy` was developed in the Autonomy Lab of Simon Fraser University, and is a fork of AR-Drone Brown driver created by Brown University. These characteristics result in high fidelity INS/GPS and camera sensor data as well as a realistic dynamic model of the group of Ar.Drone quadrotors, and a straightforward implementation of hardware-in-the-loop analysis for a quick and smooth transition into full hardware demonstrations.

Figure 1 shows the front-end of the upgraded simulator that allows for multiple quadrotors to fly simultaneously and interact with each other. Every quadrotor has a simulated inertial measurement unit (IMU) with specifications similar to those of the commercial units and a simulated GPS receiver with user-defined error specifications. The individual quadrotors use those sensors for the estimation of navigation parameters, implementing sensor-fusion techniques developed in [9]. In particular, gyroscopes provide angular velocity and by integration the attitude is estimated. As errors accumulate during integration, gyroscopes by themselves are not enough for absolute orientation measurement. Accelerometers complement this data by comparing the earth gravitational field with the quadrotor reference frame and thus giving an absolute roll and pitch measurement. The fusion algorithm is given by a Madgwick complementary filter [10] which uses the high-bandwidth of the gyroscope together with the low-bandwidth of the accelerometer to estimate orientation.

Position estimation of the quadrotor is achieved by fusing dead-reckoning with GPS measurements from a low-cost receiver. Due to the non-linear dynamics, an Extended Kalman Filter [11] is used to fuse these

two measurements.

The quadrotor 3-D models were modified with the addition of a colored marker (magenta hat) that can be used by the vehicles to visually localize each other in the formation in order to develop vision-based integrated navigation methods of UAV platoons [12].

IV. Formation definition

In order to verify the functionality of the simulator, a multirobot formation is defined for a group of 3 aerial robots, each with 4 degrees of freedom (DOF), specifically, (x, y, z, θ). This is applicable to robots such as stabilized quadrotors (UAVs).

The formation is defined following the Cluster Space Control framework [13]. This strategy conceptualizes the n-robot system as a single entity, a *cluster*, and desired motions are specified as a function of cluster attributes, such as position, orientation, and shape. These attributes guide the selection of a set of independent system state variables suitable for specification, control, and monitoring. These state variables form the system's cluster space. In this particular case, the cluster is defined with the following specifications:

- The position of the cluster (x_c, y_c, z_c) is defined by a frame located at the centroid of the formation and its orientation is given by the $(roll_c, pitch_c, yaw_c)$ angles –following the $Z-Y-X$ Euler angles convention– as shown in Figures 2 and 3.
- The shape of the formation is defined by 3 variables that specify a triangle using 2 sides and one angle. In particular, p specifies the distance between robot 1 and robot 2, q specifies the distance between robot 1 and robot 3, and β is the angle formed by robots 2-1-3.
- The orientation of the robots with respect to the orientation of the cluster is given by the variables ϕ_i, where $i = 1, 2, 3$.

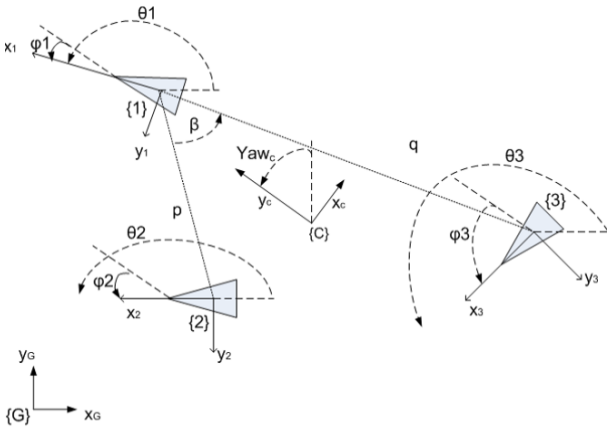

Fig. 2. Cluster parameters definition. Overhead view.

Given this definition, a set of forward and inverse kinematic equations can be written to relate the robot

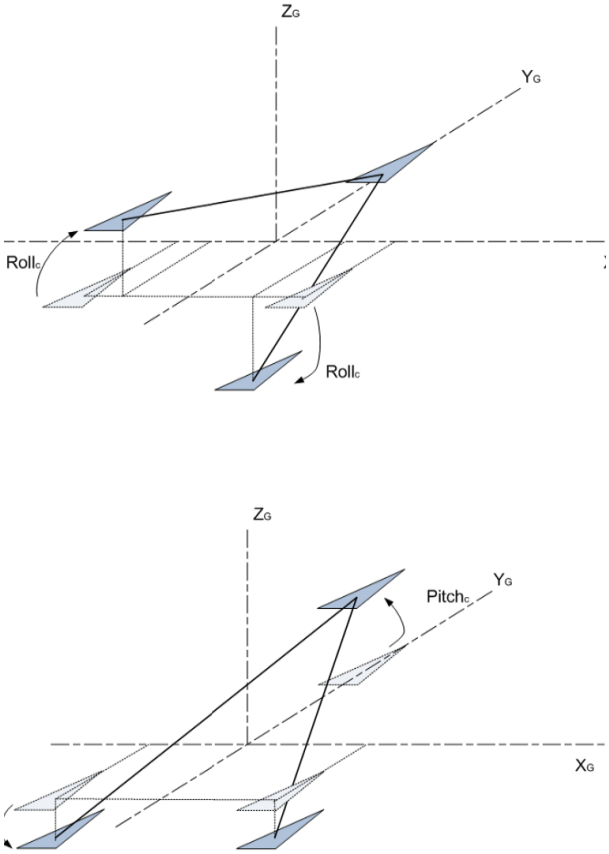

Fig. 3. Cluster parameters definition showing $roll_c$ and $pitch_c$ variables.

space variables

$$r = (x_1, y_1, z_1, \theta_1, x_2, y_2, z_2, \theta_2, x_3, y_3, z_3, \theta_3)^T, \quad (1)$$

with the cluster space variables

$$c = (x_c, y_c, z_c, roll_c, pitch_c, yaw_c, \phi_1, \phi_2, \phi_3, p, q, \beta)^T, \quad (2)$$

such that $c_i = f_i(r)$ and $r_i = g_i(c)$. Furthermore, velocity kinematics can be derived to obtain the cluster jacobian matrix that relates velocities in both spaces, such that $\dot{c} = J(r)\dot{r}$.

V. UAV Formation Controller

Figure 4 shows the architecture of the formation controller implementation. The controller operates in the space of the formation –cluster space–, which allows for an intuitive specification of the formation trajectories. Control commands are computed in cluster space where trajectories for each of the formation variables can be followed. The inverse jacobian matrix converts the resulting cluster space velocity commands to robot space velocity commands that are then applied to the vehicles in the system. Robot sensor information is converted into cluster space through the jacobian and forward kinematic relationships.

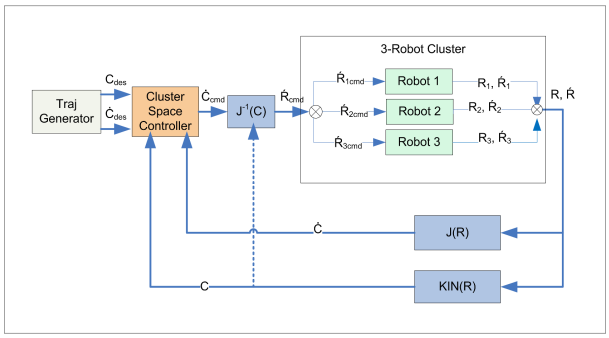

Fig. 4. Cluster space controller block diagram . Solid lines indicate signals and dotted lines indicate parameter passing.

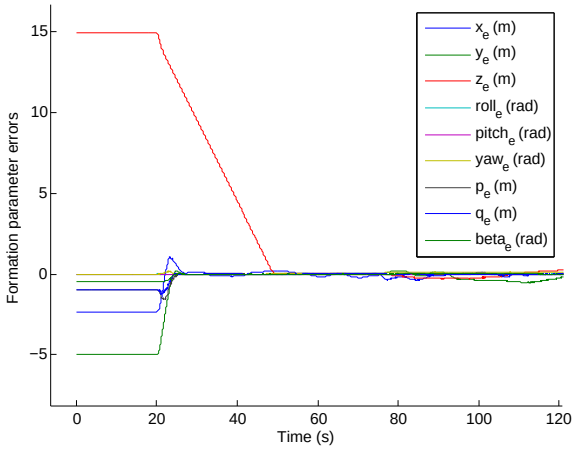

Fig. 5. Formation parameters errors starting on the ground and taking off at $t = 20\,\mathrm{s}$.

TABLE I
STANDARD DEVIATION OF THE FORMATION PARAMETERS FOR TEST 1 AFTER STEADY STATE IS REACHED.

Formation parameters' std errors	
x_c (m)	0.11
y_c (m)	0.22
z_c (m)	0.13
$roll_c$ (rad)	0.004
$pitch_c$ (rad)	0.0025
yaw_c (rad)	0.044
p (m)	0. 032
q (m)	0.035
β (rad)	0.012

VI. RESULTS

The performance of the system is evaluated implementing a proportional controller with the architecture presented in Section V. For these test cases, the accelerometers std errors are set to $0.35\,\mathrm{g}$ and the gyroscopes std errors are set to $0.01\,^\circ/s$. The GPS receiver is simulated as Gaussian with zero mean and standard deviation of $2.0\,\mathrm{m}$ (std.). The onboard cameras were not used during these tests.

The controller, as well as the formation kinematic transformations, were implemented in Python for native execution in the ROS environment.

Two test cases are presented. In the first one, three quadrotors start on the ground and take off at time $t = 20\,\mathrm{s}$. The cluster controller then regulates the formation to reach a desired final position given by the cluster parameters

$$c_{des} = (5, -5, 15, 0, 0, \pi/2, 0, 0, 0, 4, 4, \pi/4)^T. \quad (3)$$

The errors of the different cluster parameters are shown in Figure 5. It can be noted the slow dynamics of the quadrotors altitude as they climb from $0\,\mathrm{m}$ to $15\,\mathrm{m}$. The standard deviations of the formation parameters' error signals once the steady state is reached are shown in Table I.

For the second test case, the formation follows a cluster space trajectory. The input to the controller is a reference trajectory for each formation parameter:

$$\begin{aligned} x_{\mathrm{c\,cmd}}(t) &= 5\cos(2\pi f t) \\ y_{\mathrm{c\,cmd}}(t) &= -5(1 + \frac{1}{2}\sin(4\pi f t)) \\ z_{\mathrm{c\,cmd}}(t) &= 15 + \frac{5}{2}(1 - \cos(3\pi f t)) \\ yaw_{\mathrm{c\,cmd}}(t) &= 0.1t \\ p_{\mathrm{c\,cmd}}(t) &= 4 \\ q_{\mathrm{c\,cmd}}(t) &= 4 \\ \beta_{\mathrm{c\,cmd}}(t) &= \pi/4 \end{aligned} \quad (4)$$

where $f = 0.01\,\mathrm{Hz}$ is the trajectory frequency. The remaining variables are regulated at a constant zero value.

The results are shown in Figure 6 for the formation position, in Figure 7 for the formation orientation, and in Figure 8 for the formation shape parameters.

It can be seen that the formation parameters follow closely the desired trajectories. The uncertainty in the robots' yaw estimation by the sensor fusion algorithm –due to the lack of an absolute measurement of this magnitude– affects the x_c and y_c variables, specially at low velocities as can be seen in Figure 6 as well as the shape of the formation as seen in Figure 8.

Next steps include further verification of the software and setting up a hardware experimental system to contrast the simulator results with Ar.Drone hardware experimental tests.

VII. CONCLUSION

This article presented an open-source simulator for multiple Ar.Drone quadrotors that runs under the ROS operating system. The simulator includes dynamic models of the commercially available vehicles and models of their onboard sensors, together with a simulated GPS receiver. The system functionality was demonstrated by the implementation of a formation control architecture, and results for a step response and a trajectory following test were shown.

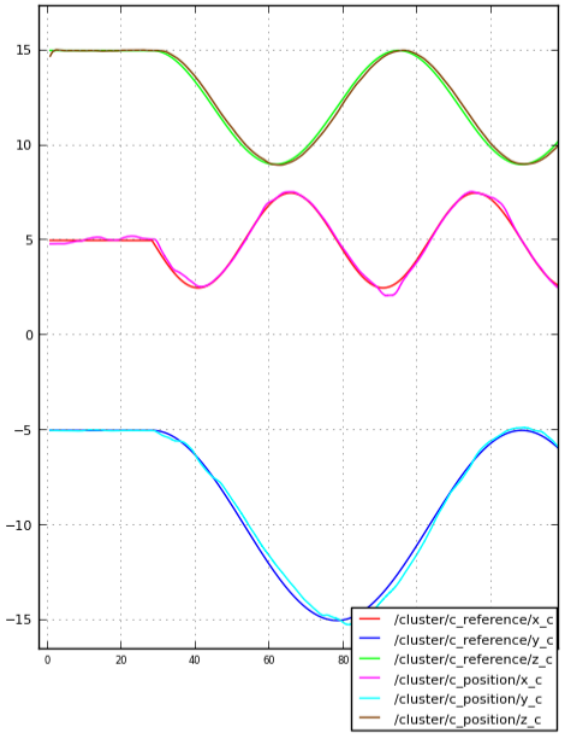

Fig. 6. Trajectory following results showing the desired and measured values of the x_c, y_c, z_c formation variables.

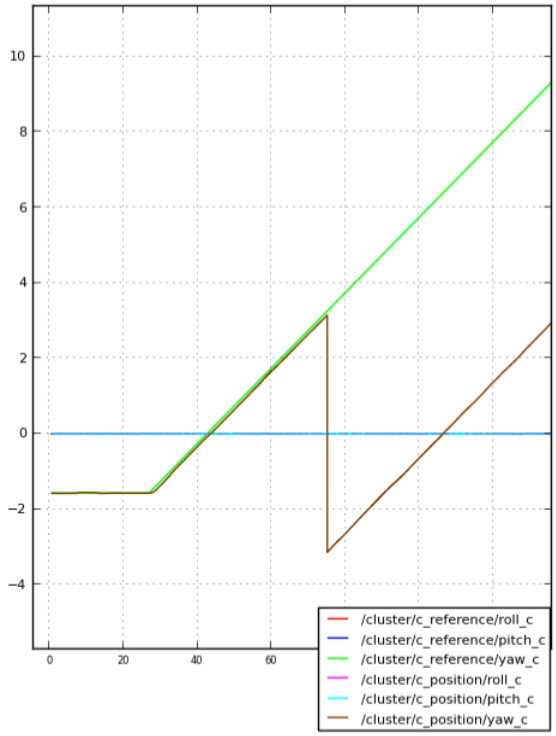

Fig. 7. Trajectory following results showing the desired and measured values of the $roll_c, pitch_c, yaw_c$ formation variables.

The interface of the simulator with the ROS environment is similar to that of the open-source driver typically used to connect through ROS with the real vehicles. This feature allows for a seamless transition between simulation, *hardware-in-the-loop* testing, and full hardware experimentation.

This work is a contribution to the overall objective of creating an integral multi-UAV testbed, consisting of a simulation environment and a hardware system, that can provide a simple and fast development platform to study guidance, navigation and control of platoons of autonomous aerial vehicles, with emphasis on formation control methods, coordinated navigation and the early demonstration of potential applications.

ACKNOWLEDGMENT

This work has been sponsored through a variety of funding sources to include USAIT Grant W911NF-14-1-0008, and the grants "Iniciación a la Investigación y el Desarrollo Tecnológico", and ITBACyT 2013-17, Instituto Tecnologico de Buenos Aires (ITBA), Argentina.

REFERENCES

[1] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar, "The grasp multiple micro-uav testbed," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 3, pp. 56–65, 2010.

[2] Mario Valenti, Brett Bethke, Gaston Fiore, Jonathan P How, and Eric Feron, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO*, 2006, vol. 63, p. 64.

[3] H Jin Kim and David H Shim, "A flight control system for aerial robots: algorithms and experiments," *Control Engineering Practice*, vol. 11, no. 12, pp. 1389–1400, 2003.

[4] Timothy W McLain and Randal W Beard, "Unmanned air vehicle testbed for cooperative control experiments," in *American Control Conference, 2004. Proceedings of the 2004*. IEEE, 2004, vol. 6, pp. 5327–5331.

[5] Rodney Teo, Jung Soon Jang, and Claire J Tomlin, "Automated multiple uav flight-the stanford dragonfly uav program," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*. IEEE, 2004, vol. 4, pp. 4268–4273.

[6] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1642–1648.

[7] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009, vol. 3.

[8] Pierre-Jean Bristeau, François Callou, David Vissière, Nicolas Petit, et al., "The navigation and control technology inside the ar. drone micro uav," in *18th IFAC World Congress*, 2011, vol. 18, pp. 1477–1484.

[9] S. Curi, I. Mas, and R. Sanchez Peña, "Autonomous flight of a commercial quadrotor," *Latin America Transactions, IEEE*, vol. 12, no. 5, pp. 853–858, Aug 2014.

[10] S Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Report x-io and University of Bristol (UK)*, 2010.

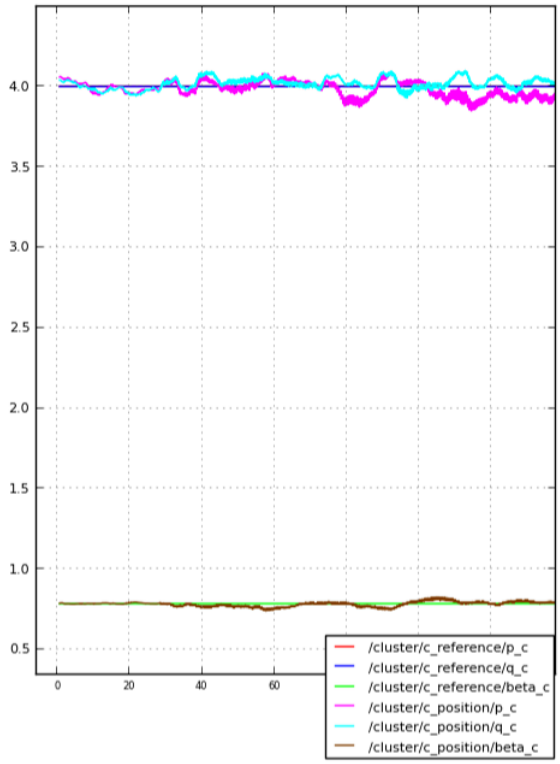

Fig. 8. Trajectory following results showing the desired and measured values of the p_c, q_c, β_c formation variables.

[11] Greg Welch and Gary Bishop, "An introduction to the kalman filter," 1995.

[12] J. Giribet, I. Mas, and R. Sanchez Peña, "Navegación integrada con visión de múltiples UAV," *Congreso Argentino de Tecnología Espacial*, May 2013.

[13] C. A. Kitts and I. Mas, "Cluster space specification and control of mobile multirobot systems," *Mechatronics, IEEE/ASME Transactions on*, vol. 14, no. 2, pp. 207–218, April 2009.