Noname manuscript No.
(will be inserted by the editor)

A Reliable Open-Source System Architecture for the Fast Designing and Prototyping of Autonomous Multi-UAV Systems: Simulation and Experimentation

Jose Luis Sanchez-Lopez · Jesús Pestana · Paloma de la Puente · Pascual Campoy

Received: date / Accepted: date

Abstract During the process of design and development of an autonomous Multi-UAV System, two main problems appear. The first one is the difficulty of designing all the modules and behaviors of the aerial multi-robot system. The second one is the difficulty of having an autonomous prototype of the system for the developers that allows to test the performance of each module even in an early stage of the project.

These two problems motivate this paper. A multipurpose system architecture for autonomous multi-UAV platforms is presented. This versatile system architecture can be used by the system designers as a template when developing their own systems. The proposed system architecture is general enough to be used in a wide range of applications, as demonstrated in the paper. This system architecture aims to be a reference for all designers.

Additionally, to allow for the fast prototyping of autonomous multi-aerial systems, an Open Source framework based on the previously defined system architecture is introduced. It allows developers to have a flight proven multi-

The authors would like to thank the Consejo Superior de Investigaciones Cientificas (CSIC) of Spain for the JAE-Predoctoral scholarships of two of the authors, as well as the Spanish Ministry for Education, Culture and Sports for funding the international research stay of one of the authors, as well as and the Spanish Ministry of Science MICYT DPI2010-20751-C02-01 for project funding.

Jose Luis Sanchez-Lopez · Jesús Pestana · Pascual Campoy
Computer Vision Group,
Centre for Automation and Robotics, CSIC-UPM.
Calle Jose Gutierrz Abascal, 2; 28006 Madrid (Spain)
E-mail: {jl.sanchez, pascual.campoy}@upm.es
www.vision4uav.eu

Paloma de la Puente
Institute of Automation and Control,
Vienna University of Technology.

Pascual Campoy
Robotics Institute - TUDelft

aerial system ready to use, so that they can test their algorithms even in an early stage of the project. The implementation of this framework, introduced in the paper with the name of "CVG Quadrotor Swarm", which has also the advantages of being modular and compatible with different aerial platforms, can be found at `https://github.com/Vision4UAV/cvg_quadrotor_swarm` with a consistent catalog of available modules. The good performance of this framework is demonstrated in the paper by choosing a basic instance of it and carrying out simulation and experimental tests whose results are summarized and discussed in this paper.

Keywords Aerial Robotics · Distributed Robot Systems · Multi-Robot Coordination · System Architecture · Open-Source · Visual Navigation · Quadrotor · Mobile Robots · Remotely Operated Vehicles · MAV

1 Introduction

Research and development, not only scientific but also commercial, related to Unmanned Aerial Vehicles (UAV) has experienced a big growth in the last few years. Although fixed-wing UAVs dominate the market of civilian and military applications, rotary-wing UAVs, with multirotor platforms being the most used ones, are becoming a good alternative for small environments tasks such as aerial photography, inspection, surveillance and search and rescue. Additionally, due to the limitations of multirotor platforms (like their small payload or its limited endurance), multirobot systems are becoming more and more popular to efficiently solve complex missions.

The design and development of an autonomous system is a hard task that requires sufficiently experienced designers to consider, during the design process, all the modules needed for its correct autonomous performance. If the goal is to create an autonomous multirobot system, the complexity increases. Once the system has been correctly designed, the development stage begins. A frequent problem for developers is the difficulty of testing the performance of their algorithms in a real flying system, because they usually do not have access to the full autonomous prototype until all the modules are in an advanced development stage.

These two problems -the difficulty of designing an aerial multirobot system and the difficulty of having an autonomous prototype of the system to test the performance of each module even in an early stage of the project- are the motivations of this paper.

In section 2, a multipurpose system architecture for autonomous multi-UAV platforms operation is presented. This versatile system architecture can be used by the system designers as a template when developing their own systems. The proposed system architecture is general enough to be used in a wide range of applications, as shown in section 3 by its successful usage in three very different projects.

To allow the fast prototyping of autonomous multi-aerial systems, we have developed an Open Source framework based on the previously defined system

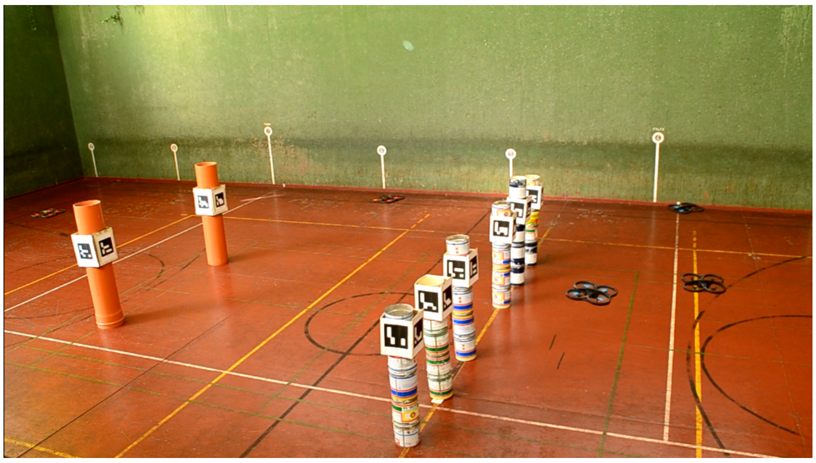

Fig. 1 Experimental flight of a basic instance of the framework "CVG Quadrotor Swarm". This basic instance is characterized by swarm behavior, using 5 AR Drones. The row of columns represents a virtual wall with a single 1.5 m opening in the middle. In order to simplify the localization problems the columns are marked using ArUco visual markers [5]. In this flight the mission of the swarm is to cross from one side to the other, while each swarm agent also has to avoid collision with the columns and the other drones. This test was designed to showcase the capabilities of this basic instance of the framework.

architecture, which allows developers to have a flight proven multi-aerial system ready to use so that they can test their algorithms even in an early stage of the project. This framework, which we call "CVG Quadrotor Swarm"and also has the advantages of being modular and compatible with multiple aerial platforms, is presented in section 4. To demonstrate the correct performance of this framework, we have chosen a basic instance of the framework and we have completed simulation and experimental tests (see figure 1), whose results are shown in sections 5 and 6. Finally, section 7 lists a set of possible future works and section 8 concludes the paper.

2 Autonomous Multi-UAV System Architecture

The first contribution of this paper is the design of an Autonomous Multi-UAV System Architecture general enough to be used for every autonomous multi-UAV system developer.

As the interest on UAVs has experienced a big growing in the last years, there exist multiple research groups working on this field. Most of them focus their researches on specific topics like perception, control, intelligence, behaviors, etc., being unable to develop a fully functional Autonomous Unmanned Aerial System. Some others, work in the field of multi-robot system, simplifying problems like localization, perception, etc., focusing on multi-robots behaviors and interaction.

Some examples are the following (see “Publications” in each group’s web page to have a full list of publications):

- Robotics and Perception Group [1] of University of Zürich (Zürich, Switzerland), led by Prof. Dr. Davide Scaramuzza.
- Vijay Kumar Lab [2] of University of Pennsylvania (PH, USA), led by Prof. Dr. Vijay Kumar (See [11]).
- Computer Vision Group [3] of Technical University of Munich (Munich, Germany), led by Prof. Dr. Daniel Cremens.
- The ETH Zürich (Zürich, Switzerland), has three very active research groups:
  Computer Vision and Geometry Group (CVG) [4], led by Prof. Dr. Marc Pollefeys.
  Autonomous Systems Lab (ASL) [5], led by Prof. Dr. Roland Siegwart, with the “SFly” project [6]
  Institute for Dynamic Systems and Control [7], led by Prof. Dr. R. D’Andrea, with the Flying Machine Arena [8].
- Heterogeneous Cooperating Teams of Robots (HECTOR) of Technische Universitt Darmstadt (Darmstadt, Germany).
- Cognitive Systems [9] of University of Tubingen (Tbingen, Germany).
- Autonomous Robotics and Human-Machine Systems (HRI) [10] of Max Planck Institute for Biological Cybernetics (MPI), led by Dr. Paolo Stegagno (Tbingen, Germany).
- Research Group for Statistical and Biological Physics of Etvs Lorand University (ELTE) (Budapest, Hungary) (See [21]).
- Autonomous Mobile Robotics Group (AMOR) [11] and Laboratory for Robotics and Intelligent Control Systems (LARICS) [12] of University of Zagreb.
- Autonomous mini-UAVS [13] of UPMC-CNRS, led by Pascal Morin (France).
- Systems Control and Flight Dynamics [14] of ONERA (France).
- Autonomous Intelligent Systems [15] of University of Bonn (Bonn, Germany)
- Aerial Vision Group [16] of Technical University of Gratz (Gratz, Austria).

[1] `http://rpg.ifi.uzh.ch`
[2] `http://www.kumarrobotics.org/`
[3] `http://vision.in.tum.de`
[4] `http://www.cvg.ethz.ch/`
[5] `http://www.asl.ethz.ch/`
[6] `http://www.sfly.org/`
[7] `http://www.idsc.ethz.ch/`
[8] `http://flyingmachinearena.org/`
[9] `http://www.cogsys.cs.uni-tuebingen.de/`
[10] `http://www.kyb.mpg.de/research/dep/bu/hri/`
[11] `http://act.rasip.fer.hr/groups_amor.php`
[12] `http://larics.rasip.fer.hr/`
[13] `http://chair-uavs.isir.upmc.fr/`
[14] `http://www.onera.fr/en/dcsd`
[15] `http://www.ais.uni-bonn.de/`
[16] `http://aerial.icg.tugraz.at/`

– Institute of Robotics and Industrial Informatics [17] of UPC-CSIC (Barcelona, Spain).
– Micro Air Vehicle Lab [18] of Technical University of Delft (Delft, Netherlands).
– Ecole Nationale de la Aviation Civile (ENAC) (Toulouse, France).

The activity of these research groups resulted on several System Architectures with an available Open-Source Framework:

– The "asctec_mav_framework" [19], developed by ASL - ETHZ, has a special focus on autonomous navigation of Ascending Technologies Helicopters [20].
– The "hector_quadrotor" [21] framework, developed by HECTOR - TU Darmstadt (see [3]), focused on heterogeneous cooperation for search and rescue tasks.
– The "telekyb" [22] framework, developed by HRI - MPI (see [6]). This framework and architecture allows the full Autonomous Multi-UAV Navigation, but it results hard to adapt to every required use (see section 3).
– The "Paparazzi" project [23] with specific hardware and Open-Source software, is developed and used by ENAC and MAVLAV - TUDelft. This project is very focused on Low-Level Control (see 2.3.1).

Finally, there also exist, commercial open projects. The most important one is the "PX4" project [24], that is an Open-Source and Open-Hardware project which aims to promote the development of low-cost UAS.

If we focus on Open-Hardware, which objective is the development of the electronics needed to develop UAS, the three main projects are: "PixHawk" project [25]; "PX4 FMU"; and "APM \ ArduPilot".

The two main Open-Source projects, which aim is the development of software framework to achieve the autonomous navigation of UAS, with special focus on Low-Level controllers (see 2.3.1), are grouped under the "Linux Foundation Drone Code" [26]; and are: "PX4 flight stack" [27]; and "APM \ ArduPilot" [28]

To the knowledge of the authors, no other work describes a full versatile open source architecture that enables the autonomous operation of a multi-robot (i.e. several robotic agents at the same time), multiplatform (i.e. het-

[17] `http://www.iri.upc.edu/`
[18] `http://www.lr.tudelft.nl/en/cooperation/facilities/mav-laboratory/`
[19] `http://wiki.ros.org/asctec_mav_framework`
[20] `http://www.asctec.de/en/`
[21] `http://wiki.ros.org/hector_quadrotor`
[22] `http://wiki.ros.org/telekyb`
[23] `http://wiki.paparazziuav.org/wiki/Main_Page`
[24] `https://pixhawk.org/`
[25] `https://pixhawk.ethz.ch/`
[26] `https://www.dronecode.org/`
[27] `https://pixhawk.org/platforms/start`
[28] `http://ardupilot.com/`

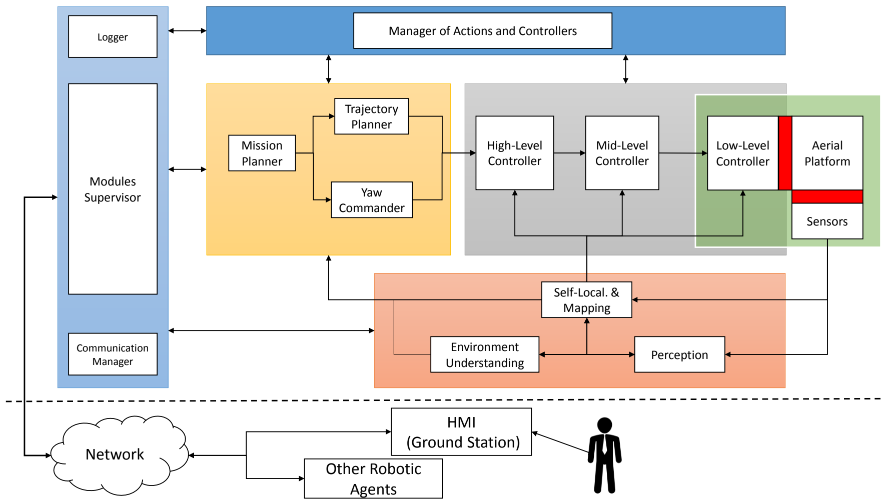

Fig. 2 Proposed Autonomous Multi-UAV System Architecture. The green block on the right symbolizes the Aerial Platform (section 2.1); the red block includes the Localization and Mapping Modules (section 2.2); the gray block covers the Control Modules (2.3); the yellow one represents the Planning and Intelligence (section 2.4); while the two blue blocks are the Supervision and Communication Modules (section 2.5). Finally, each aerial robotic agent communicates with the rest of robotic agents and with the user thanks to the Human-Machine Interface (HMI, section 2.6).

erogeneous robotic agents) system of aerial platforms (with special focus on multirotor platforms).

The versatility of the proposed architecture allows designers to use it as a template for its own aerial robotic systems regardless their complexity, being valid for the easiest cases as well as for more complex ones.

The advantages of using the proposed System Architecture are threefold: First, it allows the designers to quickly design an autonomous robotic aerial system for any application, simplifying the systems engineering stage. Second, the architecture allows the user to consider all the modules needed for the correct operation of the system, avoiding design errors due to some modules not being taken into account. Third, it allows the developers to easily focus on the working of one or several modules without ambiguity because the modules' utility is clear enough.

In figure 2, the full system architecture is represented.

2.1 Aerial Platform

In the proposed system architecture, the aerial platform represents a multirotor vehicle, but the reader could easily extend the work to other aerial platforms. The aerial platform is composed by the mechanical frame, the motors and its drives, the propellers, the battery, and a power board. The aerial platform could carry several sensors, like an Inertial Measurement Unit (IMU), cameras, an Optical Flow Sensor, altitude sensors, Global Positioning Systems (GPS) receivers, etc. Although not represented in figure 2, external sensors like Motion Capture Sensors can be utilized too with an easy extension of the proposed system architecture. Additionally, the aerial platform has a small computer (sometimes called "autopilot", "flight controller", or "flight controller unit, FCU"), that at least runs a Low-Level Controller (section 2.3.1) and a first state estimator (Self-Localization Module, section 2.2.2). The aerial platform can also carry one or more powerful computers (sometimes called "on-board computers") to run part or the full system architecture presented in this paper. In some cases, the manufactures include a Mid-Level controller (or in some cases even a High-Level controller) running on the autopilot. Additionally a basic supervisor usually runs on the autopilot.

2.2 Localization and Mapping

Localization implies being able to know the state of the drone in a map that can be previously known or unknown, using the information given by the sensors. Mapping is the action of internally generating a model of the environment with an adequate representation suitable for subsequent environment interpretation. The environment can be structured or unstructured and it can be static or it can include moving objects. The proposed architecture divides the Localization and Mapping module in three submodules as described in detail below. Additionally, each submodule can be down-sampled in several submodules, but since this is related to the application and the algorithm used, we leave this down-sampling decision for the system architecture users. Since full Localization and Mapping is a hard and a very extensive task, the three proposed submodules can be linked to each other in a bidirectional way to maximize their performance.

2.2.1 Perception

This set of modules are responsible for extracting a simpler information from the measurements given by complex sensors (like cameras, LIDAR, etc.). Some computer vision algorithms can be included here.

2.2.2 Self-Localization and Map Generation

Using the information given by the sensors and the information given by the Perception modules, the state of the drone (including its pose) can be estimated and a map can be generated. State Estimation and Simultaneous Localization and Mapping (SLAM) algorithms can be included here.

2.2.3 Environment Understanding

Once the map is built and the drone's state is estimated (so the drone is correctly localized in the map), the last step is the understanding of the environment so that this information can be properly used by other required modules. This module uses the information given by the sensors, the information given by the Perception modules and the information given by the Self-Localization and Map Generation modules to generate high-level information of the environment that the drone can easily use. State Estimation and Machine Learning algorithms can be included here.

2.3 Control

The Control modules are responsible for guiding the drone to reach the desired set-point or the command-reference. The Controllers use the information given by the Localization and Mapping module as feedback. They can be classified in three groups that are detailed in the next sub-sections depending on how close the Controller works with respect to the hardware. Apart from the Controllers, the Manager of Actions and Controllers Module (section 2.5.1) aims to allow the operation of the different Controllers that the system can have.

2.3.1 Low-Level Controller

The Low-Level Controller command references are the Thrust, the Pitch, the Roll and the derivative of the Yaw with respect to time. It interacts directly (or through a supervisor module) with the motor drives of the aerial platform.

2.3.2 Mid-Level Controller

The Mid-Level Controller is defined in this architecture as a set of controllers that transform the platform in a more usable one, adding the following features:

– Take-off and Landing maneuvers that allow the drone to start and finish a flight in an autonomous way.
– Hover maneuver that allows the drone to stay in the air in a set point without moving.
– Altitude controller that allows the High-Level controller to send height or altitude speed commands instead of Thrust commands.

Additional behaviors like Emergency can be added.

As a simplification, we can consider that the Mid-Level controller transforms the aerial platforms in a Parrot ARDrone like platform (see [2, 14]), hence simplifying the development and work with these platforms.

2.3.3 High-Level Controller

The responsibility of the High-Level Controller is the generation of Height or speed of Height, Speed of Yaw, Pitch and Roll Commands that allow the drone to follow the references given by the Planning and Intelligence Modules (section 2.4) appropriately.

2.4 Planning and Intelligence

Planning uses the information given by the Localization and Mapping modules (section 2.2) to generate command references used by the Control Modules (section 2.3) in order to accomplish a particular objective.

All underactuated multirotors have four independent degrees of freedom which are the position (x-y-z) and the Yaw angle. The proposed architecture divides the Planning and Intelligence Modules in three submodules, as explained bellow.

2.4.1 Trajectory Planner

Its goal is to generate 3D collision free trajectories (x-y-z) that can be followed by the drone.

2.4.2 Yaw Commander

The function of the Yaw Commander is the generation of Yaw angle references. This can be used with different objectives, like maximizing the performance of the on-board sensors (for example a camera or a LIDAR), or to achieve a particular goal (for example to watch over some target).

2.4.3 Mission Planner

The mission planner is the highest-level Planning and Intelligence submodule. The mission to achieve by a particular robotic agent can be defined by the user (in case of a single robot system or a swarming-type multirobot system), or can be defined by other robotic agent, usually called the coordinator (in case of a non-swarming multirobot system). The mission planner can be as complex as the mission requires.

2.5 Supervision and Communication

The Supervision and Communication Modules are key modules to ensure the correct autonomous behavior of the system, monitoring the state of all modules, activating and deactivating actions depending on its state. They are, additionally, in charge of the communication between the aerial robotic agent and the other robotic agents and the system operators. Their last mission is the logging of the state of the robotic agent so that it can be reviewed or post-processed later.

2.5.1 Manager of Actions and Controllers

This module has the function of allowing and prohibiting actions of the aerial robotic agent requested by the operator or by other modules, depending on the state of the full system. It additionally has the mission of switching properly between controllers (enabling and disabling them) without any collisions.

2.5.2 Modules Supervisor

The goal of this Supervisor is to monitor the state of each module of the system, enabling and disabling it when needed, ensuring a correct autonomous performance of the system. It also has the failure mode routines.

2.5.3 Communication Manager

This module communicates the areal robotic agent with the rest of the system, that means, with the other robotic agents and with the operators and users.

2.5.4 Logging

Finally, the Logger module, keeps a record of the events taking place on the aerial robotic agent so as to allow the users, the operators or the developers of the system to review them.

2.6 Human Machine Interface

The Human Machine Interface (HMI) allows the operators or users of the system to bi-directionally communicate with each aerial robotic agent. Both communication directions are equally important because:

– The communication from the robotic agent to the operator is needed to monitor its state and the performance of the requested mission. This will allow the operator to actuate in case of a need to redefine or abort the mission. A good user graphical interface simplifies this task.
– The communication from the operator to the robotic agent is needed if a non-rigid behavior is needed to accomplish the goal, making it possible to redefine the mission and also to start, stop and abort the mission.

3 Successful uses of the proposed Autonomous Multi-UAV System Architecture

The proposed Autonomous Multi-UAV System Architecture (section 2) has been used by the Computer Vision Group (CVG) of the Centre for Automation and Robotics (CAR, CSIC-UPM) in all its research projects since it was firstly defined in January 2013. In these nearly two years, the system architecture has been refined and converted into a fully usable and general architecture that can be used in many possible aerial robotic projects.

As a demonstration of the strength of this architecture, three different examples are shown in the following subsections.

3.1 Vision-based object following

This work is described in [17] and [15]. The goal of this work was the development of an autonomous aerial platform capable of following a user defined object using computer vision as the main source to perceive the target, and using a low-cost aerial platform, like the Parrot ARDrone, to facilitate the algorithm development. This work, requires a big simplification of the proposed System Architecture. Now, only a single aerial robotic agent is used to achieve the mission. The selected aerial platform (Parrot ARDrone, see [2, 14]) includes in its autopilot a Mid-Level Controller and a first Self-Localization Module (which estimates yaw, pitch, roll, height and speeds in x and y).

The equivalences between the proposed System Architecture and the modules described in [17] are the following:

– Perception: OpenTLD algorithm (see [10, 13]).
– Environment Understanding, and Self-Localization and Mapping: are referred to in [17] as “the estimation of the position of the target with respect to the drone, expressed in the drone’s reference frame”, using the information given by the tracker. That means, equations 1 and 2, and the left equations of figure 5 in [17].
– High-Level Controller: both “Position Controller” and “Model Identification” (to transform from speed commands to Pitch and Roll commands).
– Yaw Commander: is hidden in the equations 1 and 2 in [17].
– Supervisor: implements the proposed logic when the target is lost.
– HMI: allows the user to control the mission and to select the target to follow.

There is no Trajectory Planner or Mission Planner, because the mission is defined manually by the user, who directly interacts with the Supervisor.

3.2 2013 International Micro-Aerial Vehicle Competition

This work is described in [16] and [20]. This work successfully achieved the first award in the Indoors Autonomy Challenge of the 2013 International Micro Aerial Competition (IMAV 2013), see [9]. In this project a swarm of Parrot ARDrone aerial platforms had to complete a mission in a structured environment. There was no coordinator in the swarm and the coordination emerged as the result of the interaction of the intelligence of each swarm agent.

As detailed in section 3.1, the aerial platform includes a Mid and Low-Level controller and a first sub-module for Self-Localization.

The equivalences between the proposed System Architecture and the modules described in [16] are the following:

- Perception: “Aruco Eye”.
- Self-Localization and Mapping (second sub-module): “Pose Estimator”.
- Self-Localization and Mapping (third sub-module): “Localization and Mapping”.
- Environment Understanding: “Obstacle Generator”.
- High-Level Controller: “Trajectory Controller”.
- Trajectory Planner: “Trajectory Planner and Collision Avoidance”.
- Yaw Commander: “Yaw Commander”.
- Mission Planner: “Mission Planner”.
- Supervisor and Communication: “Hypothalamus”.

Even though not described in [16], the system also has a Logger and a HMI.

3.3 2014 International Aerial Robotics Competition

This work is described in [18]. This work successfully achieved two special awards: “Best Obstacle Avoidance Award” and “Best Trajectory Controller” in the 7th Mission of the 2014 International Aerial Robotics Competition (IARC 2014), in the Asia Venue, see [8]. In this project, a single autonomous aerial platform (an Asctec Pelican, see [1]) had to complete a “shepherding” task, guiding ten ground robots (iRobots) moving on an arena with four moving obstacles. The “shepherding” task was carried out by interacting with the ground robots (touching or blocking them). The selected aerial platform has a Low-Level controller running on-board its autopilot.

The equivalences between the proposed System Architecture and the modules described in [18] are the following:

- Perception: here should be included all the “Computer Vision Algorithms”.
- Self-Localization and Mapping (first sub-module): “Odometry-based State Estimator”.
- Self-Localization and Mapping (second sub-module): “Localization and Mapping”.
- Environment Understanding: It is included in “Localization and Mapping” and comes to being able to identify the iRobots and obstacles in the map, trying to predict their movement.
- Mid-Level Controller: “Multirrotor Driver”

- High-Level Controller: “Flight Controller” which includes several flying modes and actions.
- Trajectory Planner: Included in the “Flight Controller”.
- Yaw Commander: Included in the “Flight Controller”.
- Mission Planner: “Mission Planner”.
- Supervisor and Communication: It is divided in the “System Supervisor”; included also in the “Flight Controller” that switches between control modes; and included also in the “Multirrotor Driver” for actions.
- HMI: “Human Machine Interface”.

A Logger was also included but not mentioned in [18].

4 An Open Source Framework for Autonomous Multi-UAV Systems: CVG Quadrotor Swarm

The second contribution of the paper is the presentation of an open source and flight-proven framework that allows its users to start working easily with a simple autonomous multi-aerial robotic system designed according to the system architecture proposed in section 2.

This work was initially presented in [19] and here the work is continued by adding new features to the framework and more tests, both under simulation and real experimental ones (sections 5 and 6).

The presented framework has been designed to accelerate the prototyping of successful multi-aerial-robot behaviors for the research and development of civilian applications of small Unmanned Aerial Vehicles (sUAVs). The motivation of this framework is to allow the developers to focus on their own research by decoupling the development of dependent modules, leading to a more cost-effective progress in the project. In order to achieve this, the framework has been made public and open-source, offering several open-ended modules required for experimental multi-aerial-robot navigation.

The main design specifications of the system are the following:

- Modularity, to allow code reuse for different solutions. For this reason, the Robot Operating System (ROS) is used as middleware between modules and across computers. To better understand the characteristics of ROS, see [22, 4]. Additionally, ROS is a world wide used software middleware used by roboticists.
- Compatibility with various quadrotor platforms through the usage of a well specified interface.
- Capability of realizing multi-aerial-robot missions. The robots would be connected through WLAN, see figure 3, and they would communicate under the ROS middleware.
- Flight-proven and capability of running simulations on big parts of the developed architectures.
- Open-source, so that we can share our work and other developers can reuse any part of the architecture in their own projects.

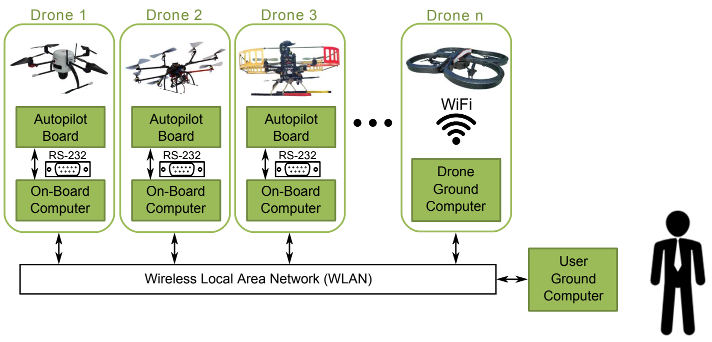

Fig. 3 The multi-robot system is composed by robotic agents, which consist of a quadrotor platform and an specific instance of the software architecture. The drone is either commanded via WiFi from a ground station or from an on-board computer. All the computers can communicate with each other through a (Wireless) Local Area Network (LAN). The communications between modules and robotic agents is implemented creating a single ROS network.

The key characteristic of our framework is modularity. This allows to create independent modules with specific functionalities which can be exploited once connected to the rest of the architecture. This modularity allows the individual testing of modules, easing the project progress. Also, understanding the modules as input-output systems permits to test in simulation the compatibility of their interfaces with the full system instance at hand.

Our framework uses the Robot Operative System (ROS) [22], a worldwide used API that eases the management of communication between the software modules of our system. The framework is fully compatible with the last ROS version (ROS Indigo) and its new building tool Catkin. The framework modules were programmed in C++ and Python. In general, the ROS communication interface has been separated from the main functionality of the modules by means of wrappers.

The compatibility with various quadrotor platforms is achieved thanks to the modularity requirement. Since each module is defined by its interface, different platforms can be used with the only requirement of respecting the specified interface. However, if a platform is heterogeneous with respect to every supported platform (i.e. it uses different sensors), the developer can leverage from the modularity requirement to minimize the required work of interfacing it with the rest of the architecture. To achieve this goal, the interface between modules has been specified and each robot agent uses its own configuration files. Each module can be executed in an on-board computer or in ground computers. However, all the computers have to be connected in a local area network (LAN) or in a wireless local area network (WLAN), see figure 3.

The framework is fully operative, which is shown in the paper through simulations and real flight tests of up to 5 drones, and was demonstrated in section 3 through three different successful projects.

Since we trust our system and we believe in sharing within the scientific community, we decided to make our framework open-source. This way, anyone interested in working with multi-robot aerial applications can use our framework as a starting point for their research and as a tool to test their own algorithms.

To share the framework with the scientific community and to develop new modules, keeping track of the past state of the development, we use a Git repository for the framework structure that links to several git repositories (one per package or module) using the tool Git Submodule. This Git tool helps in making the framework more modular. The link to the framework's code Git repository is specified in the following website: `https://github.com/Vision4UAV/cvg_quadrotor_swarm`.

Now that the main features of the framework have been presented, we proceed to describe its available modules of packages, which correspond to the modules developed for the projects cited in section 3.

The following notation is used in the rest of the section: Package is a set of code (C++ or Python) with a unitary meaning and utility. It can contain a library and / or an executable. All packages in the framework are declared as ROS packages. A Stack is a set of packages that are grouped because they have a similar meaning or utility or their meaning or utility is linked. A Module is a ROS node inside a package that adds a specific behavior or function to the drone.

4.1 Basics

This set of stacks simplify the usage of the framework and the development of new modules or packages.

4.1.1 Framework Core

This stack is a group of ROS packages which include the ROS messages and ROS services used to communicate the stack modules, and the definition of a key C++ class called "DroneModule". The "DroneModule" class has two main functions. Thanks to this class, a rigid structure is defined for the creation of new modules. This not only helps the user with the development of new modules, but also allows the supervisor module to manage and control all the modules in an easy and efficient way by means of a common interface.

4.1.2 Libraries and Utilities

These stacks include some basic C++ packages that are useful for the developing of new modules. They include interfaces for C++ libraries like pugixml[29], or newmat11[30]; a library to build an Extended Kalman Filter (EKF) (lib_cvgekf);

[29] `http://pugixml.org/`

[30] `http://www.robertnz.net/nm11x.htm`

libraries to help the user to work easily with different reference frames (lib_pose and referenceFrames); a library to create threads (lib_cvgthreads) without using the C++11 standard that is not compatible with ROS (ROS uses the C++03 standard); and a library with a mix of other utilities and control blocks (lib_cvgutils), like filters, PIDs, ...

4.1.3 Logging

Thanks to this stack, the user is able to log the state of the full system easily by configuring which topics need to be logged. The key differences between this logger and a standard rosbag file are the following: All the topics are stored in a plain text file with the time stamp and the name of the module and topic that published it, which is readable by the user. Additionally, images are stored as .png files. This log is additionally readable by Matlab thanks to a script that can be found in the framework. Additionally, it could be read as a rosbag play because every logged event is timestamped (but this functionally is not implemented yet).

4.2 Drone and Sensor Drivers

This stack includes both the interfaces (here called drivers) between the framework and the aerial platforms and the interfaces between the framework and the sensors used onboard the platforms.

The available aerial platforms are: Parrot ARDrone ([2, 14]), AscTec Pelican ([1]), and Mikrokopter Okto ([12]).

The available sensors are: px4flow[31], and uEye usb Cameras[32].

Although the framework includes a basic set of aerial platforms and sensors that allow the user to start prototyping, including new platforms or sensors is not a very heavy task, which shows the power of being able to use all the modules of the framework but only requiring the creation of an interface using the message types defined in the framework.

4.3 Localization and Mapping

The perception of the ArUco visual markers used for the IMAV 2013 Indoors Competition and the perception developed for the IARC 2014 Competition are available and included in the perception module.

The framework counts with an EKF odometry pose estimator that highly relies on optical flow; and an EKF-SLAM that relies on the ArUco visual markers used for the IMAV 2013 Indoors Competition as the set of Self-Localization and Map Generation.

[31] `http://pixhawk.org/modules/px4flow`
[32] `http://en.ids-imaging.com/`

Finally, as part of the Environment Understanding, the framework includes two modules. One of them, which was used for the IMAV 2013 Indoors Competition, transforms the map generated with the ArUco visual markers into usable obstacles like poles or walls. The second module, used for the IARC 2014 Competition, tracks the pose of the ground robots in the arena.

4.4 Control

The framework does not include any Low-Level Controller, being the manufacturer's ones the ones used for each platform.

The framework presented in this paper includes a Mid-Level Controller that works for the multirotor aerial platforms. This controller has to be tunned for each platform by using configuration files. The available flight modes are: trajectory, position, speed and object following.

A High-Level Controller that -after a tunning process- works for multirotor aerial platforms, is included in the framework. This High-Level controller provides four modes: Trajectory, Position, Speed and Object Tracking.

4.5 Planning and Intelligence

The presented framework only includes a 2D Trajectory Planner (x-y), being the extension to 3D a candidate for future work.

The framework relies on a basic Yaw Commander that allows the user to define a looking point or a looking direction.

A sequential Mission Planner allows the operator to define a mission as a set of separate tasks which are, in turn, fully described by a set of numeric parameters. The mission definition requires an xml file where the mission is defined. It has different available tasks such as: take off, land, hover, sleep or move.

4.6 Supervision and Communication

The framework includes two supervisors, the one used for the IMAV 2013 Competition and the one used for the IARC 2014 Competition.

4.7 HMI

The framework includes several nodes that act like HMI. There is a terminal user interface and several interfaces which use RViz[33] to visualize the state of the aerial robotic agents.

[33] http://wiki.ros.org/rviz

4.8 Simulators

Before a real multi-aerial-robot flight takes place, all the hardware needs to be checked and set up, including not only the quadrotor platforms but also the test environment, the external visualization and processing computers, the WLAN, etc. Another very important requirement, for security reasons, is to have one emergency pilot per quadrotor. All this means that a lot of time and people are required to set up everything for an experimental test, which also means that every test has a significant cost. Since preparing a multi-aerial-robot system for a flight is costly, real tests should preferably be carried out after the system is tested in simulation.

Thanks to the modularity of the proposed framework, it is possible to replace actual modules with simulated counterparts, since they only need to somewhat mimic the original module's interface. The value of such simulations is that they allow to test sets of modules of the actual flight architecture, easing the testing and debugging software development processes.

Several simulators are included in the framework. The reader must note that the simulators are not intended to be a very accurate representation of the drone, and they should not be used for tunning the controllers or for learning models. Therefore, the simulators do not allow to thoroughly test anything related directly to the dynamics of the multicopter. However, these simulators have proved to be big time savers using them for testing the complete architecture before a real flight.

5 CVG Quadrotor Swarm: Simulation Results

In order to demonstrate the functionality of the presented framework, some general modules were selected from the list of available ones, generating a basic instance of the framework. Once this basic instance was defined, some simulation and experimental tests were carried out. In this section the simulation results are detailed, while in section 6 the results from real experiments are shown. The simulators used in this section do not represent the accurate expected behavior of the drones (dynamic model of the drone, models of the sensors, etc.), but allow to demonstrate the functionality and mission specifications of the complete proposed system before a real flight.

The basic instance of the framework is defined in [19] and this section and the following are a continuation of this work.

The following sections of the paper present the execution of two different missions which have been called: “The Pinball” and “The Hole”. It is noted that these two simulator modules were not designed to provide an accurate behavior compared to their real counterparts' behavior. The goal was to test the rest of the system in order to debug it and improve it, and also to check the mission specifications before experimental flights. The reader is invited to visit the research group's webpage to watch videos of these simulations and the corresponding experimental flights: `http://www.vision4uav.com/?q=quadrotor_stack`.

In order to interpret the simulation and experimental flight figures, the modules specifications, which are described in [19], have to be taken into account. From them, the overall expected behavior of the drones is:

– Each drone follows a sequence of waypoints given by its mission specification. The mission planner executes it sequentially.
– During execution, the planner will attempt to find trajectories that are collision-free. If it fails, the drone will be controlled to stay in the current position.
– If other swarm agents enter the current trajectory, the planner will detect it and will stop the drone and attempt to find a collision-free trajectory.
– If another swarm agent is on the current goal waypoint, the drone is commanded to stay in the current position. A new trajectory is planned when the goal position is again free.

5.1 Mission 1: The Pinball

This mission is carried out in a volleyball court (dimensions: 9 m x 18 m). Twelve poles (four of them with a diameter of 40 cm; and the other eight of 30 cm) are spread in the court with a distribution 1-3-4-3-1, as shown in figures 4 to 6. These poles act like obstacles in the same way as the pins in the pinball. Five drones execute a navigation mission that involves crossing the obstacles zone from one side of the court to the other. The mission specification includes the following sequence of tasks:

1. Start the architecture operation and take-off,
2. Move to a goal landing location,
3. Land and stop the architecture.

In figures 4 to 6, the trajectories followed by each drone in three simulations of the same mission are displayed. As there is no high-level intelligence that synchronizes the swarm, their behaviors are not deterministic and, thus, are different in every simulation execution:

– Simulation 1, Fig. 4: each drone followed a short and direct path, with very small rectifications, to its goal landing location.
– Simulation 2, Fig. 5: some drones followed a short and direct path, but others accomplished a very long path to avoid the other drones in the court.
– Simulation 3, Fig. 6: in this case, many drones followed a long path to their goal landing location.

Since the swarm agents plan their trajectories with no interaction with a global synchronization intelligence, the different executed paths are a demonstration of a low-intelligence swarm behavior.

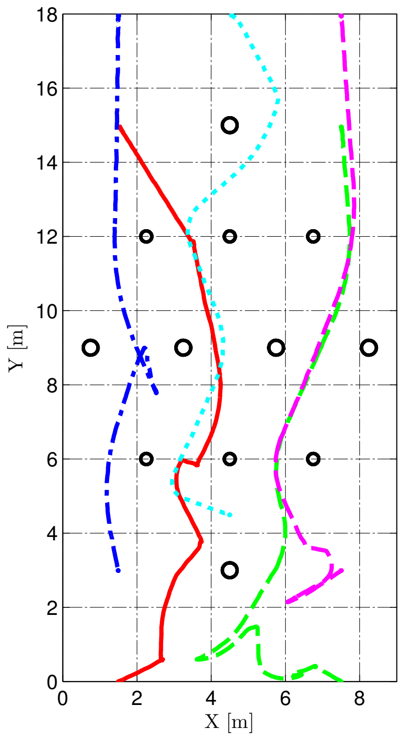

Fig. 4 Simulated flight of the Pinball Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. In this simulation, all the trajectories followed by the drones are direct and short. Some drones needed to create a longer path to avoid a collision. For example, the drone plotted in green had to go to the left of the map at the beginning of the simulation to avoid a collision with the drone in cyan. Once the collision was avoided, the drone replanned its trajectory again in order to avoid a collision with the red drone.

5.2 Mission 2: The Hole

This mission is carried out in a volleyball court (dimensions: 9 m x 18 m). Ten poles (four with a diameter of 40 cm; and the other six, 30 cm) are spread in the court with a distribution 2-6-2, as shown in figures 7 to 9. The poles in the middle create a wall obstacle with a single opening or hole whereby the drones have to cross. Five drones execute a navigation mission that requires crossing the obstacles zone from one side of the court to the other. The mission specification is defined by the following sequence of tasks:

1. Start the architecture operation and take-off,
2. Move to a goal landing location,
3. Land and stop the architecture.

In figures 7 to 9, the trajectories followed by each drone in three different simulations of the same mission are displayed. As there is no high-level intelli-

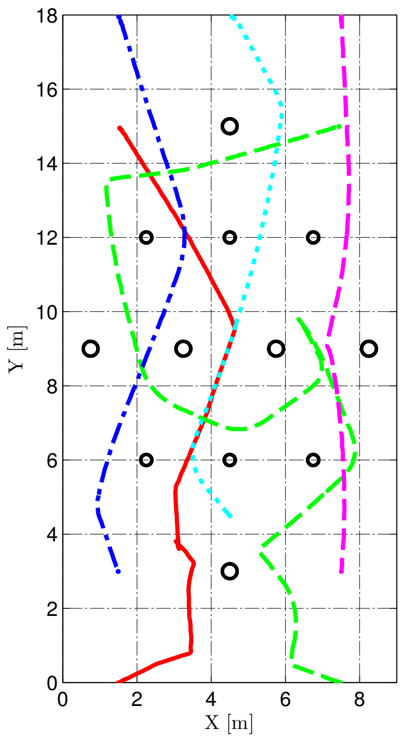

Fig. 5 Simulated flight of the Pinball Mission, where five drones flew autonomously. The figure can be interpreted similarly to Fig. 4. In this case, all the drones followed a short and direct path except the green one, which had to fly a very long path to avoid several collisions: first, with the cyan drone; and later with the magenta and red ones.

gence that synchronizes the swarm their behaviors are not deterministic and, thus, are different in every simulation execution:

- Simulation 1, Fig. 7: all the drones followed a short and direct path with very small rectifications in the trajectory.
- Simulation 2, Fig. 8: some drones followed a short and direct path, but others accomplished a longer path to avoid the other drones in the court.
- Simulation 3, Fig. 9: a simulation where some of the drones followed a longer path is shown.

Again the different executed paths are a demonstration of a low-intelligence swarm behavior.

6 CVG Quadrotor Swarm: Experimental Results

Once the system has been tested in simulation, the following step is to test it in real experiments.

As the simulations were not an accurate representation of the reality, in the real tests some issues that are not present in simulation appeared, such as:

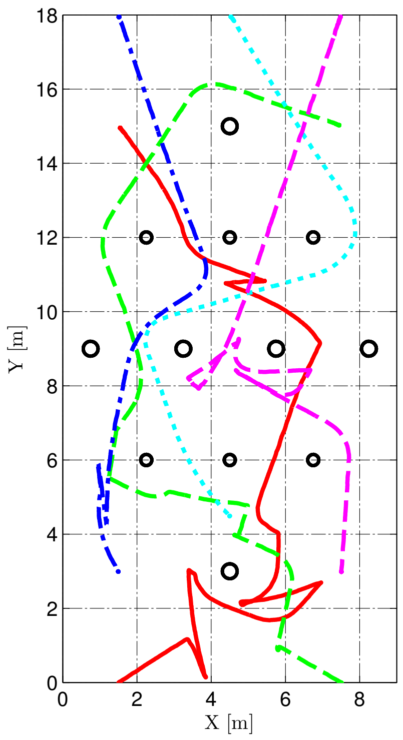

Fig. 6 Simulated flight of the Pinball Mission, where five drones flew autonomously. The figure can be interpreted similarly to Fig. 4. In this case, all the drones followed a long path to avoid collisions. These “chaotic” paths are a demonstration of a low-intelligence swarm behavior.

– Inaccurate quadrotor model
– Inaccurate drone’s sensors measurements
– Inaccurate performance of the computer vision algorithms

Those inaccuracies involve localization and control errors that the system has to minimize and deal with.

In this section, real tests of the missions simulated in section 5 are achieved.

6.1 Mission 1: The Pinball

This is the same mission defined in section 5.1. Figures 10 to 11 show the trajectories followed by each drone in two different tests. These trajectories are more noisy than the one achieved through the simulations due to the problems and inaccuracies cited above. However, the system keeps working correctly and completes the mission.

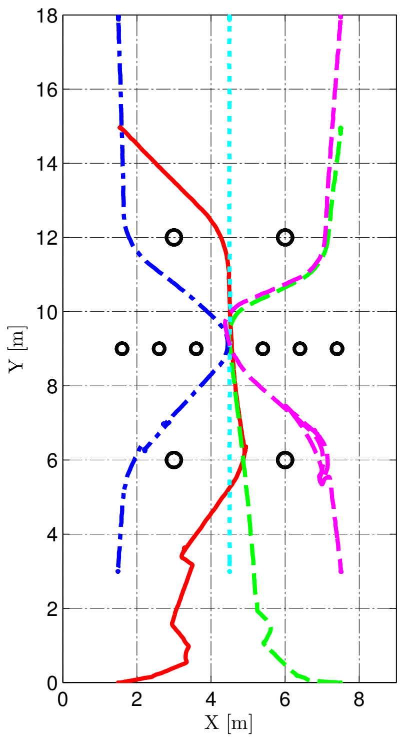

Fig. 7 Simulated flight of the Hole Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. All the drones followed a direct and short path. In the figure, it can be seen that, since there is a bottleneck in the mission, the drones had to wait while the middle passage was traversed by other drones. When there are no other drones in the hole, they planned a new trajectory.

6.2 Mission 2: The Hole

This mission is the same as the one defined in section 5.2. Figures 12 to 13 show the trajectories followed by each drone in two different tests. These trajectories are more noisy than the one achieved through the simulations due to the problems and inaccuracies cited above. However, the system keeps working correctly and completes the mission.

7 Future Work

The future work can be structured according to the two contributions of the paper.

As future work related to the autonomous multi-UAV system architecture defined in this paper, we can increase the complexity of the proposed system architecture by explicitly adding modules that determine the interaction with

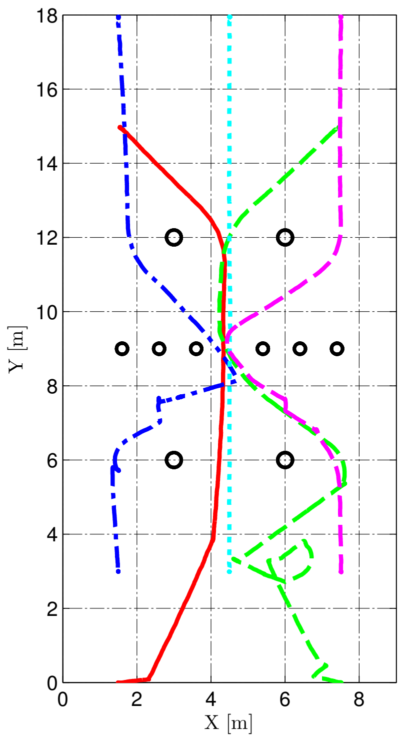

Fig. 8 Simulated flight of the Hole Mission, where five drones flew autonomously. The figure can be interpreted similarly to Fig. 7. All the drones followed a direct and short paths, including the waiting and re-planning due to the hole, except the drone plotted in green which moved in the map executing a longer path.

other robots or with the environment; or by adding modules to define the operation of actuators controlled by the system architecture (for example the aerial robots can carry on a pan and tilt camera or a robotic arm).

The possible future work related to the presented framework, the "CVG Quadrotor Swarm", is wider. On one hand, a better and simplified module integration and development can be done as future work, including the simplification of the Modules Supervisor that can be auto-generated once the modules are defined. On the other hand, new modules can be developed to have a longer catalog with more complex behaviors. Thanks to the modularity of the system, specialists in different robotics fields can contribute to specific modules of the architecture. Since the system is open-source and it is fully working both in simulation and in reality, each specialists can download it and develop a module for the system that attracts their interest and, afterwards, they will be able to test it against the rest of the system in simulation or in experimental flights. Some possible working lines are described in the following paragraphs:

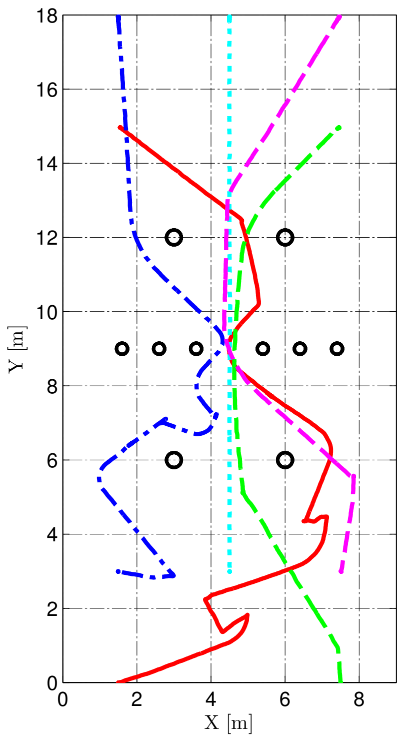

Fig. 9 Simulated flight of the Hole Mission, where five drones flew autonomously. The figure can be interpreted similarly to Fig. 7. Some drones like the magenta, green and cyan ones followed short and direct paths; but others, like the blue and the red ones followed long paths trying to find a way to traverse the hole.

One possibility is to use the described modules to conduct research in the fields of swarming and multi-robot systems. More complex behaviors could be added to the agents of the swarm in order to accomplish more complex missions that require a higher level of coordination. Or, putting behind the swarming aspect, a central unit that coordinates the robots can be added in order to improve their synchronization.

Another possibility is to carry out research in the field of fault tolerant systems, integrating the proposed architecture with a meta-control architecture that is able to detect events such as component failures (i.e. one of our modules), and after its occurrence attempt to rearrange the module architecture excluding the malfunctioning module, like in [7].

Although the system works correctly, it can only perform navigation tasks and it still requires to fly in an structured environment. This means that the mission planner could be redesigned in order to obtain a more versatile swarm behavior. Better perception, localization and mapping algorithms and sense and avoid algorithms that do not rely on visual markers could further extend the applicability of the framework.

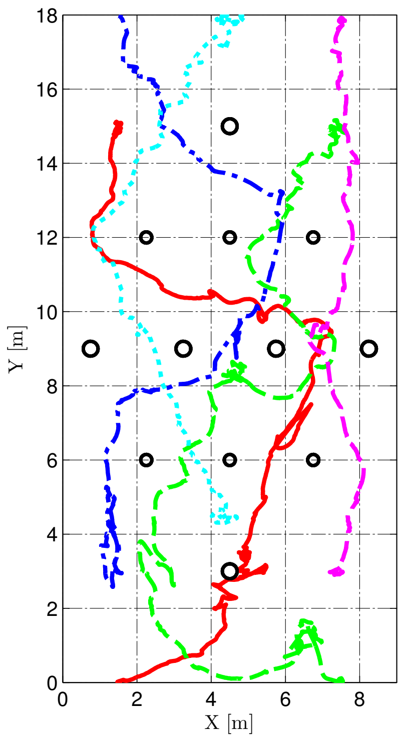

Fig. 10 Experimental flight of the Pinball Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. Some of the drones followed a direct and short path (blue, magenta, and cyan) and others a longer one (red and green). The experimental trajectories are noisier than the simulated ones, due to limitations in the simulator.

8 Conclusions

We identified two main problems during the process of designing and developing autonomous Multi-UAV Systems. In the first place, it is difficult to design an aerial multirobot system taking into account all the complex issues that have to be tackled. Secondly, it is hard to have access to a working prototype of the system so that developers can test the performance of individual modules in an integrated manner even in early stages of the project. This paper provides contributions related to these two problems.

To begin with, a multipurpose system architecture for autonomous multi-UAV platforms was presented. This versatile system architecture was the result of intense analysis and discussions, and it can be used by other system designers as a template when developing their own systems. The proposed system architecture is general enough to be used in a wide range of applications, as shown in the paper with three project examples. This system architecture intends to be a reference for all designers.

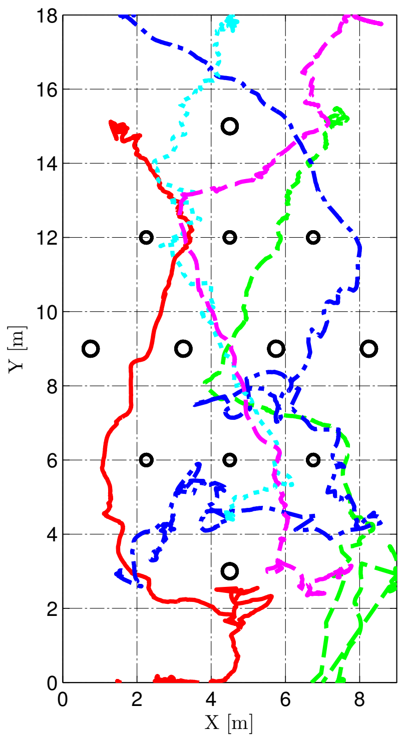

Fig. 11 Experimental flight of the Pinball Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. In this case, all the drones followed a long path. The experimental trajectories are noisier than the simulated ones, due to limitations in the simulator.

Additionally, to provide the infrastructure for fast prototyping of autonomous multi-aerial systems, an Open Source framework based on the previously defined system architecture was introduced. The “CVG Quadrotor Swarm” framework allows developers to have a flight proven multi-aerial system ready to use. This way, different algorithms can be easily tested without the need to implement a whole initial working system. This framework, which also presents the important advantages of being modular and compatible with different aerial platforms, can be found at `https://github.com/Vision4UAV/cvg_quadrotor_swarm` with a comprehensive list of available modules. The good performance of the framework has been shown by choosing a basic instance of it and carrying out simulation and experimental tests for different missions.

References

1. Ascending-Technologies (2014) Asctec pelican. `http://www.asctec.de/en/uav-uas-drone-products/asctec-pelican/`

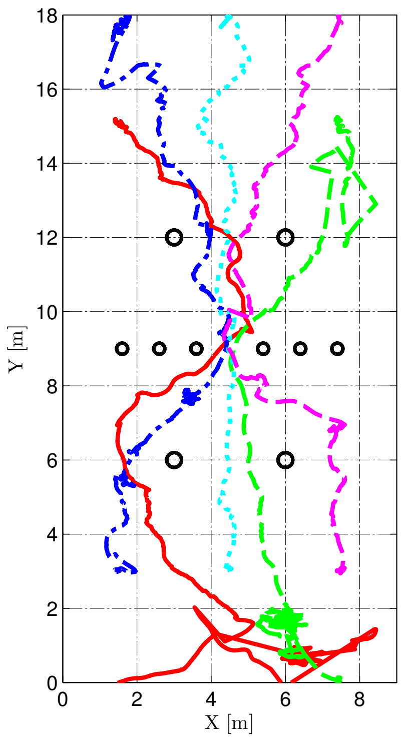

Fig. 12 Experimental flight of the Hole Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. Some of the drones followed a direct and short path (magenta, blue and cyan) and others a longer one (green and specially the red).

2. Bristeau PJ, Callou F, Vissière D, Petit N, et al (2011) The navigation and control technology inside the ar.drone micro uav. In: 18th IFAC World Congress, Milano, Italy, vol 18, pp 1477–1484
3. Darmstadt H, Kohlbrecher S, Meyer J, Graber T, Kurowski K, von Stryk O, Klingauf U (2014) Robocuprescue 2014-robot league team. RoboCupRescue 2014
4. Elkady A, Sobh T (2012) Robotics middleware: A comprehensive literature survey and attribute-based bibliography. Journal of Robotics 2012
5. Garrido-Jurado S, Muoz-Salinas R, Madrid-Cuevas F, Marn-Jimnez M (2014) Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition 47(6):2280 – 2292, DOI http://dx.doi.org/10.1016/j.patcog.2014.01.005, URL `http://www.sciencedirect.com/science/article/pii/S0031320314000235`
6. Grabe V, Riedel M, Bulthoff H, Giordano P, Franchi A (2013) The telekyb framework for a modular and extendible ros-based quadrotor control. In: Mobile Robots (ECMR), 2013 European Conference on, pp 19–25, DOI 10.1109/ECMR.2013.6698814

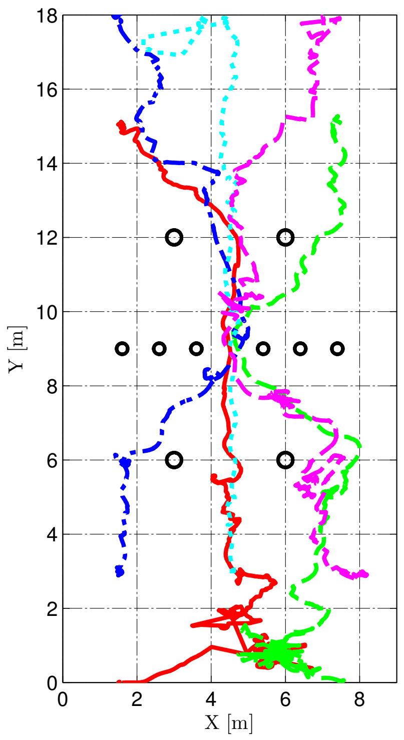

Fig. 13 Experimental flight of the Hole Mission, where five drones flew autonomously. The executed trajectories of the drones are shown as line plots. The black circles are the poles. Some of the drones followed a direct and short path (blue and cyan) and others a longer one (red, green and magenta).

7. Hernandez-Corbato C (2013) Model-based self-awareness patterns for autonomy
8. IARC14 (2014) Iarc 2014 web. `http://www.aerialroboticscompetition.org/`
9. IMAV13 (2013) Imav 2013 flight competition rules. `http://www.imav2013.org/index.php/information`
10. Kalal Z, Mikolajczyk K, Matas J (2012) Tracking-learning-detection. Pattern Analysis and Machine Intelligence
11. Kushleyev A, Mellinger D, Powers C, Kumar V (2013) Towards a swarm of agile micro quadrotors. Autonomous Robots 35(4):287–300, DOI 10.1007/s10514-013-9349-9, URL `http://dx.doi.org/10.1007/s10514-013-9349-9`
12. Mikrokopter (2014) Mikrokopter okto. `http://www.mikrokopter.de/en/products/flightsystems`
13. Nebehay G (2012) Robust object tracking based on Tracking-Learning-detection. Master's thesis, Faculty of Informatics, TU Vienna
14. Parrot (2014) Parrot ardrone 2.0 web. `http://ardrone2.parrot.com/`

15. Pestana J, Sanchez-Lopez J, Campoy P, Saripalli S (2013) Vision based gps-denied object tracking and following for unmanned aerial vehicles. In: Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on, pp 1–6, DOI 10.1109/SSRR.2013.6719359
16. Pestana J, Sanchez-Lopez J, de la Puente P, Carrio A, Campoy P (2014) A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, pp 617–622, DOI 10.1109/ICUAS.2014.6842305
17. Pestana J, Sanchez-Lopez J, Saripalli S, Campoy P (2014) Computer vision based general object following for gps-denied multirotor unmanned vehicles. In: American Control Conference (ACC), 2014, pp 1886–1891, DOI 10.1109/ACC.2014.6858831
18. Pestana J, Sanchez-Lopez J, Suarez-Fernandez R, Collumeau J, Campoy P, Martin-Cristobal J, Molina M, De Lope J, Maravall D (2014) A vision based aerial robot solution for the iarc 2014 by the technical university of madrid. In: 2014 International Aerial Robotics Competition
19. Sanchez-Lopez J, Pestana J, de la Puente P, Suarez-Fernandez R, Campoy P (2014) A system for the design and development of vision-based multi-robot quadrotor swarms. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, pp 640–648, DOI 10.1109/ICUAS.2014.6842308
20. Sanchez-Lopez JL, Pestana J, de la Puente P, Carrio A, Campoy P (2014) Visual quadrotor swarm for the imav 2013 indoor competition. In: ROBOT2013: First Iberian Robotics Conference, vol 253, Springer International Publishing, pp 55–63
21. Vásárhelyi G, Virágh C, Somorjai G, Tarcai N, Szörényi T, Nepusz T, Vicsek T (2014) Outdoor flocking and formation flight with autonomous aerial robots. IROS 2014
22. Willow-Garage (2014) Ros: Robot operating system. `www.ros.org/`