Rapid Development and Prototyping Environment for Testing of Unmanned Aerial Vehicles

Tiago Alexandre da Silva Oliveira
tiago.alexandre.oliveira@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

January 2021

Abstract

In this dissertation, an indoor multi-vehicle rapid prototyping platform is designed and implemented at the ISR Flying Arena, to support the development and testing of control and navigation solutions for unmanned aerial vehicles. The hardware architecture devised for the prototyping environment comprises: i) an optical motion capture system providing vehicle position and attitude ground-truth; ii) a set of offboard computers managing communication between systems and running user programs; and iii) multiple quadrotors. In order to provide abstraction of the vehicles and to automate the communication between systems through reliable protocols, a set of software modules were programmed using an object-oriented approach. These modules relieve the user from implementing low-level flight routines and communication tasks. An additional group of software tools was also created to allow offboard flight logging and monitoring. With the purpose of enabling testing of the deployed algorithms before experiments with physical vehicles, a fully configurable and easy-to-use simulation environment, including a solution to emulate a motion capture system, was also developed. The devised setup allows for a mixed environment of physical and simulated quadcopters, extending testing to conditions that are physically unfeasible at the ISR Flying Arena. In the end, several control solutions, including a formation-control algorithm, were deployed and tested, validating the adopted architecture and showcasing its robustness and scalability. The created prototyping platform is a key enabler of future research and education in aerial robotics, having already been used in the experimental validation process performed within the scope of the MSc Theses of other students.

Keywords: ISR Flying Arena, Multirotors, UAV Testing Environment, Aerial Robotics, GNC

1. Introduction

The use and demand of Unmanned Aerial Vehicles (UAVs) have been rapidly increasing across different industries due to their autonomous flying capabilities, onboard decision making power, ability to communicate primary and mission-related information in real-time, possibility to carry different types of payloads, and their aptness to perform hazardous tasks in a vast range of scenarios [1]. Consequently, the research and development of mission-oriented UAVs has grown at an accelerated rate, fueled by consistent advances in air-frame materials, propulsion systems, avionics, sensors, and power sources.

To support the development of mission-oriented UAVs and to ease academic research of cutting-edge guidance, navigation, and control solutions, universities are designing indoor multi-vehicle testbeds that surpass weather and daylight constraints. Iterative motion learning [2] and cooperative ball throwing and catching [3] are just a few examples of scientific breakthroughs that were conducted in and benefited from these dedicated spaces.

The first impactful indoor UAV-oriented testbed documented in the literature was the MIT RAVEN (Real-time indoor Autonomous Vehicle ENvironment) that enables the rapid prototyping of navigation and control algorithms of different types of vehicles, such as autonomous multicopters, fixed-wing drones, and ground-based rovers [4]. An experiment involving the three types of vehicles can be managed by only a single operator, presenting a substantially reduced logistical cost compared to an outdoor test. However, the system has a limit of 10 simultaneous vehicles. This testbed also allows control of the environmental conditions of the indoor space, that can range between ideal to wind induced.

Another well documented aerial robotics platform is the ETH Zurich's Flying Machine Arena [5]. The Flying Machine Arena architecture is presented in Fig. 1. The position and attitude measurements of the vehicles are provided by the indoor optical motion capture system, typically working at 200Hz, to a station of ground computers. Here, the user

code module runs estimation and control algorithms that generate flight commands. Finally, a copilot mode runs a failure detection function that supervises those commands, only allowing the appropriate ones to be sent to the vehicles. All communication is done in a high-frequency, asynchronous way that guarantees nonexistence of delays due to retransmission attempts. To overcome some inconveniences of the absence of synchronization, all sensor data is time-stamped against local hardware clocks.

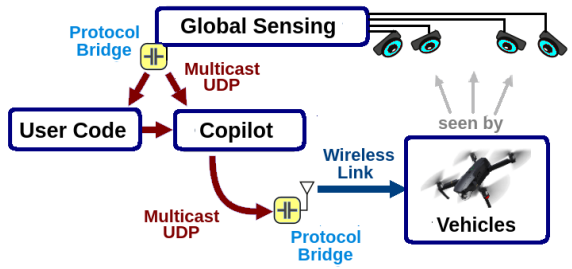

Figure 1: Flying Machine Arena architecture.

Despite being very versatile and crucial for scientific and technological progress in aerial robotics, UAV-devoted testbeds comprise a vast amount of dedicated software and hardware whose use requires good fundamentals of computer science, substantial knowledge on the communication protocols, and continued reconfiguration. These aspects pose a barrier to their usage that can slow or inhibit research and stagnate innovation. Therefore, the driving force behind this thesis is to encourage more research in aerial robotics and the development of more courses and laboratory classes involving UAVs by not only settling a dedicated testbed at the Institute for Systems and Robotics (ISR), equipping its researchers and students with a powerful tool for extensive testing and validation of new UAV-related algorithms, but by also eliminating its inherent working barriers, by not requiring the users to have deep Linux and programming knowledge, background in hardware, and by automating all communication between systems.

2. System Overview

The architecture devised for the developed rapid prototyping platform results from combining:

1. The fundamental design and operational mechanism of the indoor testbeds documented in the literature, that have three common components: i) aerial vehicles; ii) an optical motion capture system; and iii) ground computers. The pose of the vehicles is provided by the motion capture system to the ground computers that run navigation and control algorithms. The computed actuator commands are sent to the vehicles via wireless communications. This type of architecture has proved to be efficient by enabling the testing of innumerous scientific works. Therefore, it is used as the basis of the developed system.

2. The PX4 autopilot, an UAV autopilot system that provides abstraction from the vehicles, onboard estimators, onboard controllers, procedures for autonomous maneuvers, processes for decoding data from the sensors, algorithms for encoding data to the actuators, and an external connectivity module that eases all the communication between the vehicle and the ground computers. The PX4 autopilot is well suited for the ISR Flying Arena because: i) it is open source, and as a consequence, its firmware is publicly available, can be modified to perfectly fit the architecture of the tests platform, and can be adapted to specific tests and scenarios; ii) it is configurable, tunable, and compatible with multicopters, fixed wing aircrafts, and VTOL drones; iii) it has simulation-in-the-loop and hardware-in-the-loop capabilities that enable the simulation and thorough testing of the PX4 autopilot, all of its systems, and their interfaces with outside command software before the field tests; iv) it comes with failsafe modes that protect users and equipment whenever the position estimate is unreliable or whenever the vehicle loses connection with an offboard system; v) it communicates with exterior modules through a validated messaging protocol called MAVLink, that is available in widely used programming languages such as C++ and Python and that is compatible with a set of robotics libraries and tools called ROS (Robot Operating System); and vi) it runs a Real Time Operating System (RTOS) that guarantees that critical flight tasks are completed within a specific range of time, which ensures, for instance, that the motors are actuated in the right moments and that an up-to-date state of the vehicle is always available to the user.

3. A dual operation mode of simulation and actual physical flights. This means that user programs running in a ground computer can simultaneous communicate with a PX4 mounted on a physical vehicle and communicate with a PX4 SITL instance “mounted” on a vehicle running on a realistic simulator.

The resulting architecture is represented in Figures 2 and 3. In Fig. 2, a detailed view of the physical tests environment is displayed, whereas in Fig. 3 a detailed view of the simulation environment is depicted.

2.1. Physical tests environment

In the physical/real tests environment, the vehicles are confined to the limits of the ISR Flying Arena, an indoor facility of dimensions 7m × 4m × 3m. The arena is equipped with an Optitrack motion capture system composed by eight cameras that provide high-frequency measurements of the position and attitude of the vehicles. These measurements are transmitted to the ground computers via Ethernet. The blue dashed arrow in Fig. 2 represents the special cases when there is an onboard companion computer capable of receiving and decoding the Optitrack pose and sending it to the PX4 through a serial port.

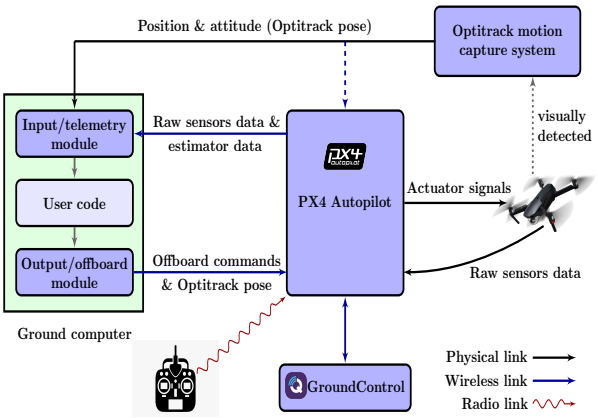

Figure 2: Developed architecture for the physical tests environment of the ISR Flying Arena.

The ground computers run offboard guidance, navigation, and control algorithms. These algorithms are implemented in the user code block. To enable the quick and effortless use of the arena and to fully automate communication between the ground computers and the remaining systems of the prototyping platform, there are two modules assisting the user code block:

- An input or telemetry module that receives and makes available to the user the position and attitude measurements from the motion capture system, along with the data provided by the PX4 Autopilot. The PX4 provides the raw measurements of the sensors installed onboard the vehicle and the output of its extended Kalman filter. With all this information, and for navigation purposes, the user can decide between implementing their own estimation algorithm or directly use the received estimates. The input or telemetry module runs in a background thread and starts to execute as soon as the ground computer establishes connection with the PX4. This module completely automates data reception and data processing.
- An output or offboard module comprising a set of methods that send instructions to the PX4, such as arming commands, takeoff requests, and attitude and thrust references. This module abstracts data sending. The user just needs to call the appropriate methods and the module will send the data to the PX4, according to the MAVLink communication protocol. Note that this module also sends to the PX4, in a background thread, the pose of the vehicle provided by the Optitrack motion capture system and processed by the input module. The PX4 uses this information as an input for its onboard estimator.

The PX4 autopilot can also receive commands from the QGroundControl, an open-source ground control station that communicates with the PX4 through the MAVLink protocol and that can also be used to monitor and modify vehicle parameters. Finally, the RC Controller functions as a safety link, enabling the user to land, disarm, and shutdown the vehicle at any moment. The implementation of the physical tests environment and the software developed to decode, process, and send information between systems is described in Section 3.

2.2. Simulation environment

The architecture of the simulation environment is presented in Fig. 3. The global system is developed with a modular design, where each component has rigorous and well-defined functions and interfaces. Consequently, the simulation environment has the same architecture and interconnections as the real environment. The only difference is that the hardware parts (the PX4 autopilot, the vehicles, and the Optitrack motion capture system) are replaced by equivalent software blocks. These blocks that change when experiments are conducted in the simulation environment, instead of the physical tests platform, are highlighted in red in Fig. 3.

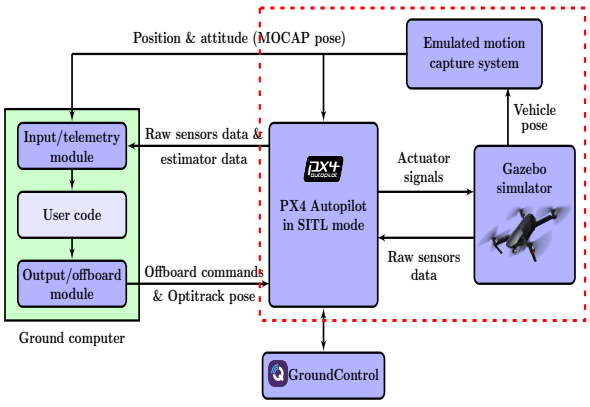

Figure 3: Developed architecture for the simulation environment.

In simulation mode, the behavior of the vehicles is computed by a simulator software, based upon their physical models. Through the actuator signals received from the PX4 SITL instance, the physics engine of the simulator computes the motion of the vehicles and the new sensor measurements. These raw measurements are transmitted back to the PX4. The position and attitude of the vehicles are retrieved from the simulator by a software script that, by adding white Gaussian noise to the retrieved quantities, emulates the measurements generated by a real motion capture system. These noisy measurements are sent to the ground computers and to the PX4. The PX4 autopilot runs in simulation due to its SITL capabilities. The ground computer runs exactly the same modules as in the real tests environment and, due to the modular design adopted, does not need to know whether it is communicating with real or simulated hardware. The implementation of the simulation environment and the emulation of the motion capture system is described in Section 4.

3. Flying Arena

The ISR Flying Arena consists in an indoor test space of dimensions 7m × 4m × 2.5m and a set of eight Optitrack motion capture cameras. Fig. 4 describes in detail the low-level architecture of the devised physical tests environment implemented at the ISR Flying Arena, that was designed according to the NED (North-East-Down) coordinate system, which is standard in aeronautical applications.

First, the Optitrack cameras detect special passive markers placed on the body of the vehicles, that reflect infrared light. Then, this tracking data is sent, via Ethernet, to a computer running the Motive software. By feeding the tracking data to its advanced solvers and to its high-level filters, the Motive computes the position and attitude of the vehicles with a positional error less than 0.3mm and a rotational error less than 0.05°. Finally, the Motive sends the position and attitude data to a router that broadcasts it to the local network. Note that the Optitrack system computes the pose data according to a ENU (East-North-Up) inertial frame.

After being broadcasted to the local network, the pose data provided by the Optitrack system needs to be decoded and processed in the ground computers and in the onboard companion computers, so user programs and the extended Kalman filter of the PX4 can fuse it with the measurements provided by the inertial sensors to produce estimates for the position and attitude of the UAV. The decoding and processing stages are implemented using the Robot Operating System (ROS) middleware and the MAVLink-Router, a library that transforms ROS topics into MAVLink streams and routes them to other endpoints, such as the PX4.

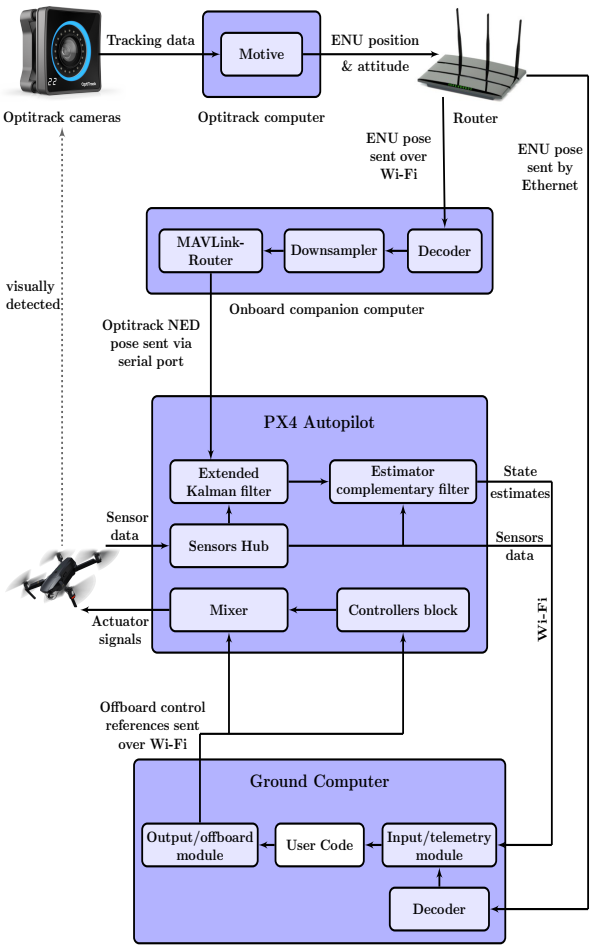

Figure 4: Flow of information between the modules of the physical tests environment, for vehicles with an onboard companion computer.

The ground computers receive the position and attitude of the vehicles, in ENU coordinates, through an Ethernet link. Then, a decoder block reads this information and publishes it into a ROS topic. The decoder block was built using the VRPN (Virtual Reality Peripheral Network) client, a ROS node that connects to the VRPN server used by the Optitrack system to stream data to the local network and exposes the information over a ROS topic. After being decoded, the position and attitude data is converted to the NED coordinate frame, the one adopted for the testing setup, and is finally made available to the user.

The companion computer receives the position and attitude data generated by the Optitrack system via Wi-Fi. The decoder block is equivalent to the one implemented in the ground computer. It was also programmed using the VPRN client that publishes the received pose information into a ROS topic. After the decoding block, the position and attitude data is submitted to a downsampling process. The Optitrack system provides positioning data to

the local network at a frequency of 180 Hz. In order to avoid exhausting the bandwidth of the communication channel, which could cause delays in the communication with the PX4 and loss of packets, the downsampling block republishes the pose data into a new ROS topic, dropping two of every three messages received. Therefore, the new ROS topic receives new position and attitude updates at a frequency of 60 Hz. Finally, by using the MAVLink-Router library, the new ROS topic is transformed into a MAVLink stream and is sent, through a serial port, to the PX4 autopilot. During this step, the position and attitude are converted from ENU coordinates, used by the Optitrack system and the ROS middleware, to NED coordinates, used by the MAVLink protocol and the PX4 autopilot. Note that the sensor measurements and the output of the extended Kalman filter of the PX4 are sent, via Wi-Fi, to the ground computer. This gives the users freedom to implement its own estimation algorithms, by fusing the position and attitude data retrieved from the Optitrack system with the sensor measurements received from the PX4, or to simply use the state estimates provided by the PX4. In the absence of an onboard companion computer, the decoder, downsampler, and MAVLink-Router modules have to run in the ground computers.

It should be noted that the subsystems of the PX4 autopilot were tuned according to the properties of the testing setup, the Optitrack motion capture system, and the vehicles. For instance, the extended Kalman filter of the PX4 was configured to use the position and yaw measurements given by the Optitrack motion capture system and to rely on the accelerometers and gyroscopes to produce low-latency and low-drift estimates of the roll and pitch angles. This is due to the fact that the inertial sensors measurements are sufficient to produce satisfying low-latency and low-drift roll and pitch estimates, but the readings of the magnetometers are disturbed by the electric motors and affected by magnetic anomalies of the indoor environment. To enable the fusion of low-latency measurements of the accelerometers and gyroscopes with the position and yaw measurements received from the Optitrack system, that reach the PX4 with some delay due to communication overhead, it was necessary to tune the extended Kalman filter with the correct time difference between the arrival of the Optitrack and the IMU measurements. A rough estimate of this delay was obtained from the logs by checking the time offset between the IMU and the Optitrack data. Then, this value was further refined by performing a set of experiments with distinct delay values, and by checking the resulting estimator innovations. The time delay obtained, 20ms, corresponds to the one that yielded the smallest innovations.

4. Simulation

Simulators allow testing of navigation and control solutions in a quick and safe way. Users can interact with a simulated vehicle just as they might with a physical one, by using the QGroundControl software or by running offboard programs on the ground computers to send commands and control references to the PX4 autopilot. Before attempting to fly actual vehicles, it is recommended to use the simulator to ensure that the estimation and control algorithms work properly and the vehicles behave as expected.

The selected simulator for the devised prototyping setup is Gazebo, a powerful 3D robotics engine suitable for testing autonomous vehicles. Gazebo was the chosen simulator because: i) it is compatible with the software-in-the-loop capabilities of the PX4 autopilot; ii) it supports all kinds of aerial vehicles, such as multicopters, fixed wing aircrafts, and VTOL drones; iii) it accepts custom models of those vehicles, so that the user can simulate UAVs with dynamic and kinematic properties close to those of the physical drones used in the ISR Flying Arena; iv) it offers plugins to simulate the behavior of sensors and actuators; v) it supports multi-vehicle simulation; and finally vi) it is compatible with the ROS middleware, a set of robotics libraries and tools used in this thesis.

Fig. 5 presents a detailed description of the software programs created to launch the simulation environment. One of the goals of this work is the development of an user interface enabling the quick and effortless use of the simulation framework. The user interface developed consists in the *gazebo.launch* file represented in Fig. 5. To launch simulations with this interface, the user just has to fill the file arguments with the desired configurations for Gazebo, for the simulated vehicles, and for the PX4 SITL instances. Then, a sequence of software programs operating in the background will automatically launch the simulation, according to the arguments requested. The *empty_world.launch* program launches Gazebo in the selected world. The *gui* argument determines whether Gazebo will run with or without the graphical user interface. The *drone.launch* program is responsible for spawning a vehicle and starting a PX4 SITL instance. For multi-vehicle simulations, multiple instances of the *drone.launch* program are executed.

The compact interface created enables users to benefit from the full capabilities of the simulation environment without prior experience with Gazebo, the ROS tools, or the PX4 firmware. Without the software programs presented in Fig. 5, users would have to extensively edit multiple different files of the PX4 firmware, which is a time consuming process, to fully configure simulations.

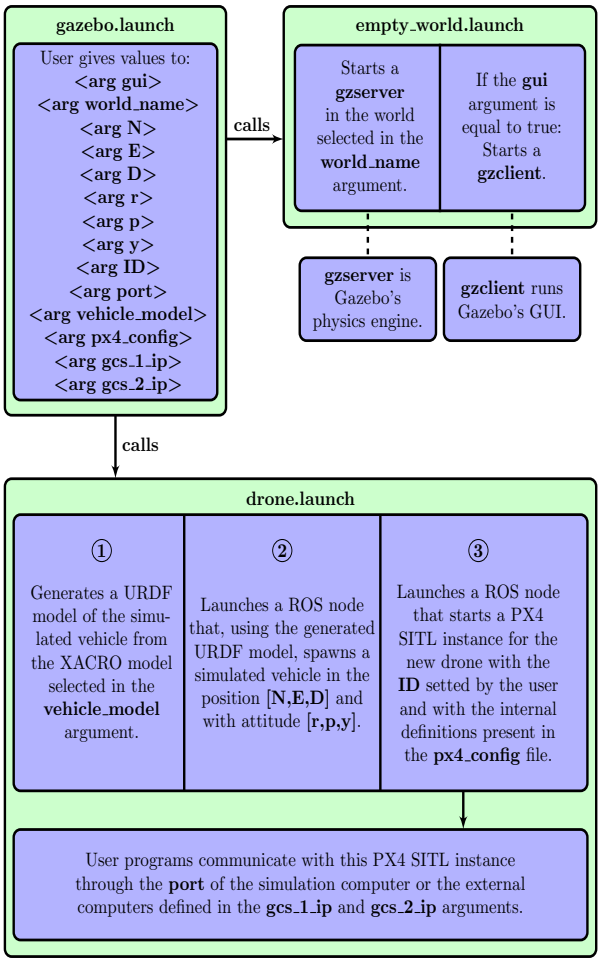

Figure 5: Description of the software programs developed to launch simulations.

In order to have a simulation environment with the same architecture as the physical tests environment of the ISR Flying Arena, it was necessary to emulate a motion capture system capable of providing, to the user programs and to the running PX4 SITL instances, high-frequency measurements of the position and attitude of the simulated UAVs. The created software program uses the capabilities of the ROS middleware to continuously retrieve the current position and attitude of the simulated drones from Gazebo. Then, after adding Gaussian white noise to the retrieved true position and attitude values (to represent the uncertainty of the measurements provided by a real motion capture system), sends the resulting measurements in the appropriate format to the correct PX4 SITL instances, while also making the data available to the user programs. Before starting experiments in the simulation environment, the user should wait for the messages indicating that the software program that emulates the motion capture system is working and that the EKF of each PX4 autopilot is successfully using the external pose measurements.

5. User Programs

As depicted in Figures 2 and 3, two modules were developed in this thesis to support user programs. The first is the input or telemetry module, that features the methods that perform the low-level tasks of subscribing to the information published by the PX4 and by the motion capture system, and of making that information available to the user in a standardized way. The other one is the output or offboard module, that stores the methods that can be called by the user programs to send offboard commands and control references to the vehicles. Since drones are physical entities represented by both data and behavior, and in order to provide abstraction and encapsulation of the vehicles, the input and output modules were programmed using an object-oriented approach. The description of the classes developed to support user programs is presented in Fig. 6.

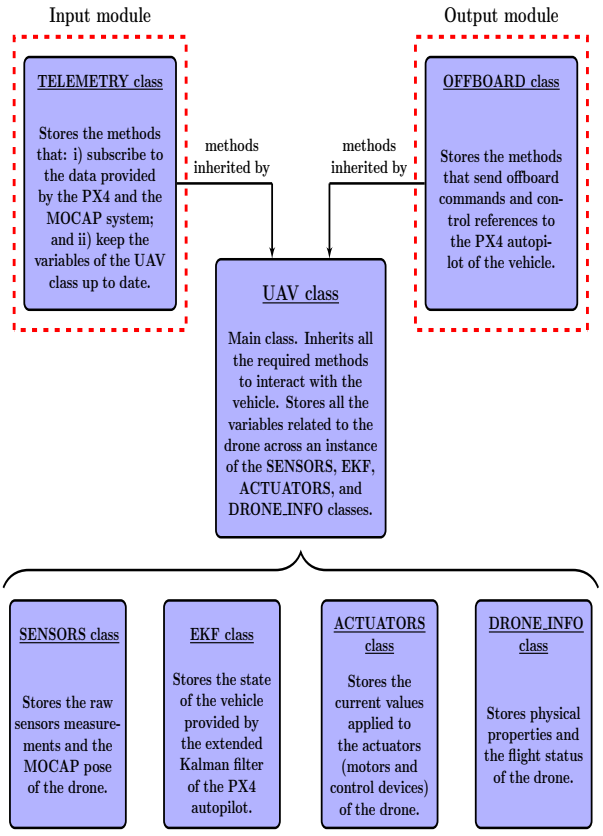

Figure 6: Description of the classes developed to support user programs.

The UAV class is the core class of the developed framework and its function is to represent a vehicle. It inherits all the methods required to receive information and send commands to the drone and it contains the variables that store the current state of the vehicle. The user only needs to create an

object of the UAV class (instead of an object of each of the other six represented classes) to have access to all the variables and methods that interact with a vehicle. This eases the use of the capabilities of the ISR Flying Arena. The input module, in the object-oriented approach adopted, corresponds to the Telemetry class. The methods of the Telemetry class, inherited and automatically called by the objects of the UAV class as soon as they are initialized, subscribe to the information published by all systems of the ISR Flying Arena and make this data accessible to the user as variables of the Sensors, EKF, Actuators, and Drone Info classes. These four classes store, in an intuitive and structured way, the set of variables that contain the data related to a vehicle. Note that there is a clear separation between the different types of data. For instance, the variables that store the raw measurements of the sensors are stored in a completely different data structure from the one that stores the state of the drone provided by the EKF of the PX4, which ensures that users always know the source of the information they are accessing. The output module, in the object-oriented approach created to support the development of user programs, corresponds to the Offboard class. This class comprises a set of methods, inherited by the UAV objects, that automate the procedure of sending data and instructions (such as arming commands, takeoff requests, and attitude and thrust references) to the PX4 autopilot of the drones. The classes presented in Fig. 6 hide the complexity of the rapid prototyping environment from the user.

It should be noted that four pairs of input and output modules were developed, one for each of the four communication libraries adopted (MAVROS C++, MAVROS Python, MAVSDK C++, and MAVSDK Python). The multiple communication solutions grant flexibility to the tests platform because some modules employ the ROS middleware, which offers valuable tools for robotics applications, while others are lightweight, which enables them to run in small onboard computers with limited resources. Researchers must employ the pair of modules created from the most advantageous communication library for their experiment.

Finally, in order to help users launch their offboard programs, which can be challenging especially when adopting the MAVROS communication libraries, a Bash program was created. This Bash program automatically starts all the required processes to perform an experiment, further reducing the difficulty involved in using the real and the simulation environments of the ISR Flying Arena. The users only have to define, in a configuration file, the physical properties of each vehicle and the user programs and tools they want to run.

5.1. Additional Tools

An additional group of software tools was also created to enable: i) emulation of sensors; ii) flight logging; iii) flight monitoring; iv) flight visualization; and v) later reproduction of the performed experiments. These tools enhance the capabilities of the prototyping platform and support the users in the validation of the GNC algorithms. These group of tools was extensively used in the tests stage described in Section 6.

6. Tests and Results

In order to validate the architecture and the software programs designed for the ISR Flying Arena, that are publicly available in [6], an extensive testing process was performed, in which several control solutions were successfully deployed.

6.1. PX4 position controller

The first experiment consisted of a multi-vehicle situation in which two drones tracked a set of desired setpoints using the internal position controller of the PX4 autopilot. This test allowed to establish a high level of confidence in the developed setup before advancing to more complex and demanding trajectories and before introducing custom controllers in the loop. It also enabled to ensure that the position controller of the PX4 is stable and that it can be used by researchers to rapidly validate navigation/estimation algorithms without having to invest time implementing a control solution. This experiment was repeated multiple times in order to test both the real and the simulation environments, and in order to test all software modules developed. Fig. 7 exhibits two drones performing the setpoints tracking test in the ISR Flying Arena.

Figure 7: Two drones performing the setpoints tracking test in the ISR Flying Arena.

Fig. 8 presents the evolution of the desired, simulated, and real altitude of the drones with time, during these setpoints tracking tests. The responses obtained in the physical and simulated environments are almost coincident, which demonstrates the importance of the simulator for a first validation of the GNC algorithms before advancing to tests with physical vehicles. The plot of Fig. 8 was automatically generated by the offboard logger tool

introduced in Section 5.1.

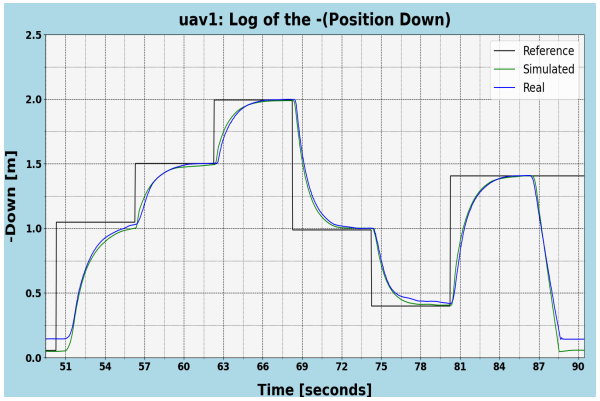

Figure 8: Evolution of the altitude of the drones with time in the setpoints tracking test.

The fact that these tests have been carried out with success, proves that the systems of the ISR Flying Arena work as designed. The position and attitude of the vehicles generated from both the Optitrack and the emulated motion capture system are indeed reaching the PX4 autopilots, that successfully merge this data with the IMU measurements. Similarly, these tests attest that user programs and the PX4 are, in fact, exchanging commands and telemetry information. These experiments help validate the architecture, the configuration process, and the programs created for both the simulation and the physical tests environments.

6.2. PID trajectory tracking controller

This section aims to demonstrate that the created setup enables the implementation of custom control solutions. For this purpose, a classic PID trajectory tracking controller was deployed, tuned, and tested in a set of fast and demanding trajectories. The implemented position controller acts as an outer loop controller, since it provides attitude and thrust references to the attitude controller of the PX4.

The trajectories employed consist of sinusoidal parametric equations known as Lissajous curves, that are frequently employed in strategies of aerial surveillance. These trajectories are also aggressive (requiring from vehicles roll and pitch angles of $\pm 20^{o}$ for successful trajectory tracking) and visually appealing so that they can be used in public demonstrations of the ISR Flying Arena. Fig. 9 exhibits a quadrotor performing a Lissajous trajectory in the ISR Flying Arena, whist Fig. 10, shows the top view of one of the performed Lissajous trajectories.

Since the experiments with the Lissajous trajectories were completed with success, it is possible to conclude that the created setup allowed to rapidly deploy and test the PID position tracking controller

Figure 9: Quadcopter performing a Lissajous trajectory in the ISR Flying Arena.

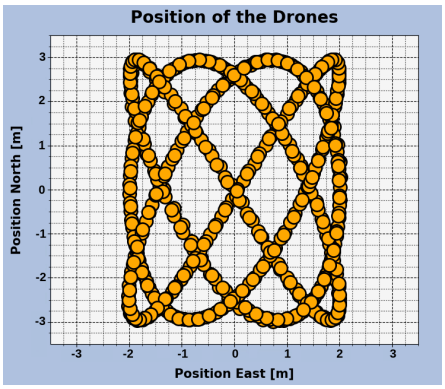

Figure 10: Top view (or North-East view) of one of the Lissajous trajectories performed.

in both the real and the simulation environments. The implementation of the controller was a fast process because, for each individual test performed, it was only necessary to create a user program with the control algorithm and the trajectory. The low-level communications tasks and the remaining systems of the ISR Flying Arena were already programmed and configured, and were easily reused.

The developed position controller was also employed and tested in slower trajectories with letter shapes, mostly intended to be reproduced in public demonstrations. Through these tests it was possible to validate the scalability of the adopted architecture. Fig. 11 showcases almost 50 vehicles performing a trajectory that spells the word FLY, in the simulation environment.

6.3. Formation-control algorithm

In order to demonstrate that the ISR Flying Arena is also suitable for testing complex multi-vehicle control solutions, a formation-control algorithm [7] was deployed and tested. The formation topology adopted is exhibited in Fig. 12. The vehicle 1 is the formation leader whereas the remaining vehicles are followers. In the devised experiment, the leader tracks a time-varying trajectory and the followers orbit around him.

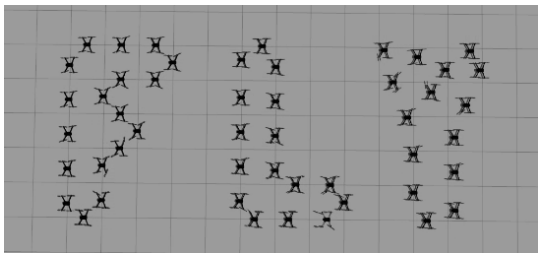

Figure 11: Set of quadcopters spelling the word FLY in the simulation environment.

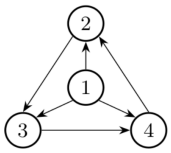

Figure 12: Adopted formation topology.

Due to equipment limitations, the formation-control algorithm was tested in a mixed environment of physical and simulated vehicles. Figures 13 and 14 show, respectively, two real and two simulated drones simultaneously performing the experiment.

Figure 13: Quadcopters 1 and 2 of the formation.

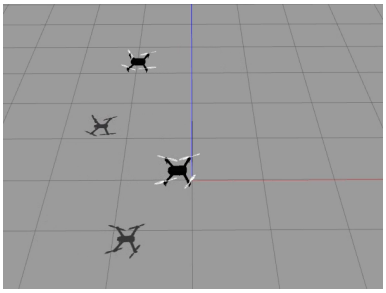

Figure 14: Quadcopters 3 and 4 of the formation.

This test demonstrates that the created setup enables the validation of GNC algorithms with part of the drones flying in the ISR Flying Arena and the other part being simulated in the Gazebo software. This mixed environment extends testing to conditions that are physically unfeasible in the arena.

During the experiments, users can monitor the four vehicles of the formation in the same graphical window using the developed Gazebo-based or Matlab visualization tools, as shown in Figures 15 and 16.

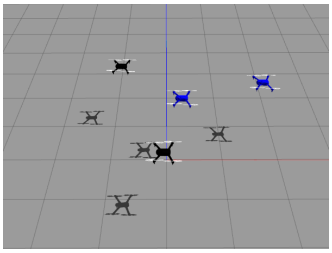

Figure 15: Gazebo-based visualization tool displaying all the complete formation in the same window.

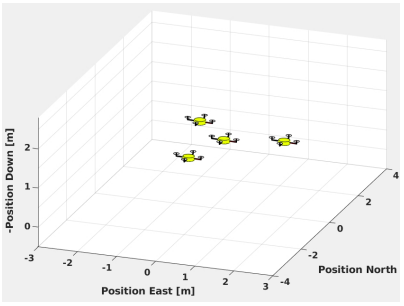

Figure 16: Matlab-based visualization tool showing all the complete formation in a single window.

Figures 17 and 18 exhibit the position results obtained in the formation-control test. The real quadcopters are represented in orange and blue. The orange vehicle is the leader and the remaining drones are the followers. Fig. 17 presents the convergence of the vehicles from their initial position to their formation position, whilst Fig. 18 proves that, after the initial convergence, the followers successfully kept the formation while orbiting around the leader.

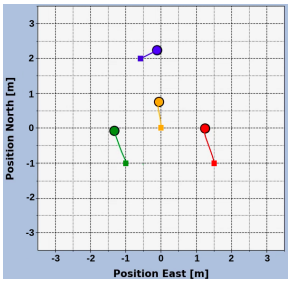

Figure 17: Formation movement at $t = 10s$.

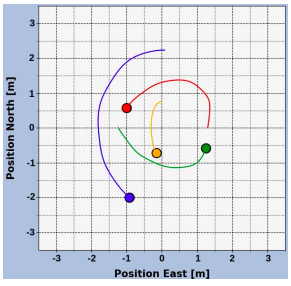

Figure 18: Formation movement at $t = 20s$.

These results match the ones obtained in the original research and prove that it is possible to successfully implement and validate formation control algorithms using the framework designed in the scope

of this thesis. Furthermore, the tests performed with the time-varying formation reinforced that the architecture adopted for the ISR Flying Arena is robust, scalable, and flexible, allowing experiments with real and simulated quadcopter interaction.

6.4. Supporting other researchers

Once the testing stage was completed and the rapid prototyping framework was deemed robust enough, it was successfully applied in the validation process of GNC solutions by other students, in the course of their theses works. This is solid evidence that the main goal of this thesis was successfully achieved.

The first research work whose testing and validation process was facilitated by the created setup consisted of a navigation system based on distance measurements [8]. By using the framework developed in this thesis, and within just two hours, the researcher was able to: i) attach an acoustic transponder to the drone; ii) create a landing site to smooth the landing of the vehicle and thus not damaging the transponder; iii) calibrate the Optitrack system; iv) create a user program for tracking the desired trajectory; v) simulate the trajectory tracking; and vi) conduct several experiments at the ISR Flying Arena with a physical vehicle. The flight logs were automatically generated by the created offboard logger. Without the support of the rapid prototyping setup developed in this thesis, the validation of the navigation algorithm in an experimental environment would be a time consuming effort.

The second thesis research assisted by the prototyping framework was related to predictive control strategies for aggressive parcel relay maneuvers using drones [9]. By using the created prototyping setup the researcher was able to immediately conduct simulations and tests with physical drones in the ISR Flying Arena using his own computer.

7. Conclusions

The goal of this dissertation was to design and implement an indoor multi-vehicle rapid prototyping platform for development and testing of guidance, navigation, and control solutions for unmanned aerial vehicles. With that in mind, an aerial robotics testbed with modular architecture was devised, tailored and implemented at the ISR Flying Arena. In order to enable the testing of the deployed algorithms before experiments with physical vehicles, a fully configurable simulation environment, featuring a solution to emulate a motion capture system, was also developed.

With the intention of providing abstraction of the vehicles and to automate all the low-level communication tasks and flight routines required to perform experiments in the ISR Flying Arena, a set of software modules were programmed, using an object-oriented approach and four different communication libraries. An offboard logger program, emulated sensors, and a set of visualization and monitoring tools were also created to further enhance the capabilities of the prototyping platform.

Finally, both the physical and the simulation prototyping environments were validated by an extensive testing process, where several control solutions, including a formation-control algorithm, were successfully deployed. These tests demonstrated that the architecture of the designed setup is robust, successfully addresses scalability, and enables experiments simultaneously involving physical and simulated vehicles, overcoming space and equipment limitations. The developed prototyping framework was also used in the experimental tests and validation process performed within the scope of the MSc Theses of two other students. Ultimately, it was proved that the rapid prototyping environment designed is a key enabler of future research and education in aerial robotics.

References

[1] R. Austin. *Unmanned Aircraft Systems: UAVS Design, Development and Deployment*. Wiley, 2010.

[2] D. Mellinger, N. Michael, and V. Kumar. Iterative learning of feed-forward corrections for high-performance tracking. *International Conference on Intelligent Robots and Systems*, 2012.

[3] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea. Cooperative quadrocopter ball throwing and catching. *International Conference on Intelligent Robots and Systems*, 2012.

[4] J. P. How and J. Teo. Adaptive Flight Control Experiments using RAVEN. *Yale Workshop on Adaptive and Learning Systems*, 2008.

[5] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea. A platform for aerial robotics research and demonstration: The Flying Machine Arena. *Mechatronics Journal, Volume 24, Issue 1*, Feb 2014.

[6] ISR Flying Arena - Digital Repository. URL `https://tiagoalexnd@bitbucket.org/dsorglobal/tiagooliveira.git`.

[7] P. Trindade, R. Cunha, and P. Batista. Distributed Formation Control of Double-Integrator Vehicles with Disturbance Rejection. *Proceedings of the 21st IFAC World Congress*, July 2020.

[8] J. Franco. Sistema de navegação baseado em medidas de distância. Master's thesis, Instituto Superior Técnico, Jan 2021.

[9] J. Pinto. Model predictive control strategies for aggressive parcel relay maneuvers using drones. Master's thesis, Instituto Superior Técnico, 2021.