electronics

MDPI

Article

Open-Source Drone Programming Course for Distance Engineering Education

José M. Cañas [1,*], Diego Martín-Martín [2], Pedro Arias [3], Julio Vega [1], David Roldán-Álvarez [1], Lía García-Pérez [4] and Jesús Fernández-Conde [1]

1 Department of Telematic Systems and Computation, Rey Juan Carlos University, 28942 Madrid, Spain; julio.vega@urjc.es (J.V.); david.roldan@urjc.es (D.R.-Á.); jesus.fernandez@urjc.es (J.F.-C.)
2 Electronic Technology Area, Rey Juan Carlos University, 28933 Madrid, Spain; diego.martin.martin@urjc.es
3 JdeRobot Organization, Alcorcón, 28922 Madrid, Spain; pedroariasperez96@gmail.com
4 Industrial Engineering Department, Francisco de Vitoria University, Pozuelo de Alarcón, 28223 Madrid, Spain; lia.garcia@ufv.es
* Correspondence: josemaria.plaza@urjc.es; Tel.: +34-914-888-755

Received: 12 November 2020; Accepted: 14 December 2020; Published: 17 December 2020

Abstract: This article presents a full course for autonomous aerial robotics inside the RoboticsAcademy framework. This "drone programming" course is open-access and ready-to-use for any teacher/student to teach/learn drone programming with it for free. The students may program diverse drones on their computers without a physical presence in this course. Unmanned aerial vehicles (UAV) applications are essentially practical, as their intelligence resides in the software part. Therefore, the proposed course emphasizes drone programming through practical learning. It comprises a collection of exercises resembling drone applications in real life, such as following a road, visual landing, and people search and rescue, including their corresponding background theory. The course has been successfully taught for five years to students from several university engineering degrees. Some exercises from the course have also been validated in three aerial robotics competitions, including an international one. RoboticsAcademy is also briefly presented in the paper. It is an open framework for distance robotics learning in engineering degrees. It has been designed as a practical complement to the typical online videos of massive open online courses (MOOCs). Its educational contents are built upon robot operating system (ROS) middleware (de facto standard in robot programming), the powerful 3D Gazebo simulator, and the widely used Python programming language. Additionally, RoboticsAcademy is a suitable tool for gamified learning and online robotics competitions, as it includes several competitive exercises and automatic assessment tools.

Keywords: distance learning; open educational platform; drone programming; gamification; Python; ROS middleware

1. Introduction

The impact of unmanned aerial vehicles (UAVs) in daily life has primarily grown in the last decade. Beyond its classic employment in aerial photography or geographic mapping, an increasing number of drone-based solutions hit the general market and make human life more comfortable every year. Drones are increasingly being used for video recordings, wildlife monitoring, precision agriculture, disaster management, entertainment, industrial inspections, etc.

UAV technology's growth has created the need to train professionals in the sector, taking the current limits further and producing new drone applications to serve the general public. UAV design is a cross-disciplinary field involving many technologies: electronics, mechanics, computer science, telecommunications, etc. Drones are composed of hardware (sensors, actuators, processors) and software. Drone programming allows more autonomous applications and widens the span of tasks that can be solved with them. A significant part of drone intelligence and value resides in its software. Therefore, drone programming abilities are increasingly demanded in the job market and increasingly taught in higher education.

Drone programming is rapidly becoming a key market for graduates in computer science and diverse university engineering degrees. Beyond classroom learning at universities, many private companies (such as Udacity, Udemy, TheConstruct, etc.) and universities themselves provide online courses to learn how to program a drone. Massive online open courses (MOOC) explore new possibilities for distance learning, taking advantage of the Internet. They typically include video classes, slides, and educational documents.

MOOCs provide useful videos explaining theory contents and drone programming foundations in distance learning, but they typically undershoot the practical side. This aspect is critical in this area, where the "learn by doing" paradigm is the most effective approach and provides the best learning results.

This article presents an open-source drone programming course based on RoboticsAcademy, an open-access educational platform for robotics distance learning. The course's primary goal is to provide an open-access tool that facilitates drones' programming in different scenarios, applying concepts related to computer vision, artificial intelligence, automation, autonomous navigation, or control algorithms. The course is ready to be used for drone programming distance learning in higher education in engineering. It provides a set of exercises covering different real-world drone applications (e.g., search and rescue, following a road or visual landing) and focusing on the practical side. Additionally, RoboticsAcademy is a suitable platform for online drone programming competitions and gamified learning, as it includes several online competitive exercises.

Typical drone systems for commercial applications are composed of a flying vehicle and a ground station. In some classic applications, drones are teleoperated by a human operator using a remote with a radio link. In contrast, they follow preplanned missions in others, sent from a ground station and composed of a sequence of waypoints. Once the task is set, the onboard autopilot can command the drone movements to automatically follow the required track using a GPS sensor, inertial measurement unit (IMU) sensors, and a fast control loop. Two well-known autopilot firmware platforms are PX4 and ArduPilot, which communicate to the ground station using the standard MAVLink protocol.

In the last few years, drone applications have been increasing their scope, and more onboard intelligence and sensors, such as cameras, are required. Industrial inspection or autonomous indoor mapping are two illustrative examples. UAVs are progressively seen as aerial robots, and therefore the most popular robot software frameworks have started to support them. For instance, the robot operating system (ROS), the de facto standard in robotics, includes the MAVLink extendable communication node for ROS package (MAVROS), which allows the command of missions and obtaining feedback from MAVLink drones. This trend democratizes access to drone programming by introducing a standard layer of abstraction and many reusable components. ROS provides a general framework to program the autonomous drone logic and facilitates integrating already developed software pieces (drivers, libraries, stacks, nodes, etc.). It undoubtedly reduces the entry barrier, also helping to migrate software from one drone to another.

Since ROS was not conceived as an educative platform, it lacks specific educative contents to be followed in the classroom or assessment tools for student's code. Another problem of using ROS as a learning tool is the wide variety of possibilities provided and its ecosystem's overwhelming size: packages, libraries, tools, drivers, etc., causing a big "learning shock" and generating frustration. It is probably the best choice for drone programming, but it is not enough on its own as an educative platform. RoboticsAcademy aims to provide such educative content using ROS as drone middleware.

The remainder of this paper is organized as follows: the next section provides a summarized review of related work. In Section 3, we discuss the main features and internal architecture of the RoboticsAcademy platform in detail. The fourth section presents the distance-learning drone programming course based on RoboticsAcademy. This platform's usage as a tool for online drone programming competitions and gamified learning is described in Section 5. Finally, Section 6 draws the main conclusions and points out future lines of research.

2. Related Work

There are two trends in teaching robotics contents at university: subjects focused on industrial robots [1–7] and subjects based on mobile robots [8–13]. In the first case, manipulation control techniques, inverse kinematics, or trajectory calculations are usually addressed. In the second case, the techniques generally explained are local and global navigation, position control (avoiding obstacles), perception, self-location, etc. This dichotomy of contents is also reflected in the robotic platforms that are used in the exercises.

Both types of robotic subjects have a notably practical bias. Student interaction with robots facilitates the learning and assimilation of the theoretical concepts, algorithms, and techniques [14]. Exercises promote the paradigm of learning through doing, that is, active learning. They usually occur within a specific robotic framework, using a teaching tool or a teaching suite where robot behavior is programmed using a particular language. A great diversity of hardware platforms is used to support the experimental component of robotics courses.

Beyond being a widespread platform in robotics education for pre-university students, LEGO MINDSTORMS robots such as NXT or EV3 have also been used in many university courses [15–17]. As recent examples, LEGO robots are used to teach motion planning [18], physical manipulation [19], and control systems [20] in real robots. These works combine LEGO robots and MATLAB.

According to Esposito [21], MATLAB and C language remain the dominant choices in programming environments. A minority uses free, open-source packages such as the robot operating system (ROS) or OpenCV.

2.1. Distance Learning in Robotics

Digitalization is changing how knowledge is created, consumed, and taught; exploring new educational possibilities such as e-learning where universities increasingly offer massive online open courses (MOOC). Universities such as Stanford, MIT, Harvard have been leading initiatives in this direction since 2011. E-learning allows reaching a mass audience, gaining visibility, and capturing talent. This movement has also influenced the robotics field. There is an increasing number of robotic courses of this style [22], such as Artificial Intelligence for Robotics [23] from Stanford University (Udacity, Sebastian Thrun) or Autonomous Mobile Robots [24] from ETH Zurich. Another example is the MOOC on the Art of Grasping and Manipulation in Robotics [25] from the University of Siena.

Virtual [6,8,14] and remote laboratories are frequent in literature. For instance, the SyRoTek teaching platform [26] provides web access to 13 small real mobile robots operating in an enclosed space—the arena. This remote laboratory was used by more than 80 students from the Czech Republic and Argentina in exercises about robot exploration, navigation, and map building. It uses the Player/Stage framework and ROS interfaces. Another interesting example of a remote laboratory is [27], where several physical robots such as CoroBot, VEX, and Pioneer, and two robotics arms such as Nao's arm and RTX Scara, were programmed or commanded from a distance. It was successfully used in an undergraduate course on cyber-physical systems.

Another distance framework for robotic arms is RobUALab [6,14], programmed in Java and web technologies, which allow the teleoperation, experimentation, and programming of a manipulator. It provides a remote laboratory with a web 3D viewer connected to a real Scorbot ER-IX arm located at Alicante University. It also includes a virtual laboratory with an industrial cell built using augmented reality, including an equivalent simulated robotic arm and as many obstacles as desired. Using this, students can test different configurations, evaluate pairs, both Cartesian and articulation trajectories, and program the arm with command lists.

This trend in web-based and practical educational frameworks also appears in other related fields. For instance, Google Colaboratory (https://colab.research.google.com) provides a simple-to-use infrastructure where the students can learn deep learning using Jupyter notebooks from their web browser, which are run on the Google servers. No installation by the students is required.

In robotics, this is complemented by using robotic simulators through a web interface. For example, one crucial online teaching initiative is Robot Ignite Academy from TheConstruct [28]. It is based on ROS, the Gazebo simulator, and Jupyter, providing several short courses. Another is RobotBenchmark (https://robotbenchmark.net) from Cyberbotics. Based on the Webots simulator, it is open and provides twelve Python exercises with several robots such as Thymio II and Nao humanoid.

Another important ROS-based initiative is Robot Programming Network [29–31]. This action extends existing remote robot laboratories with the flexibility and power to write ROS code in a web browser and run it in the robot on the server-side with a single click. It uses Moodle, VirtualBox, HTML, and Robot Web Tools as its underlying infrastructure.

Recent Internet technologies open new possibilities for distance learning in robotics. For instance, in ROSLab [32], ROS has been combined with JupyterLab technology to integrate documentation and robotics software, and Docker container technology. JupyterLab is very useful for teaching purposes, as lessons or exercises may be delivered as Jupyter notebooks. These notebooks are programmed and run locally by students, without any dependency problems, and in any operating system, just inside a container. Another relevant example is cloud computing. For instance, Amazon RoboMaker (https://aws.amazon.com/robomaker/) [33] allows cloud robotics, simulating, and deploying robotic applications at a cloud-scale, which may then be available for distance students. It provides an integrated development environment, simulation service, and fleet management capabilities.

2.2. Drone Programming Education

Beyond the courses based on industrial robots and mobile robots, there is an increasing number of drone/aerial robotics courses in higher education. Additionally, they also appear in the MOOC platforms. For instance, EdX offers the "Autonomous Navigation for Flying Robots" course [34], provided by the Technische Universität München. It is mainly focused on the camera-based navigation of a quadcopter [35,36], explaining all the underlying theoretical concepts such as linear algebra, 2D and 3D geometry, motor controllers, probabilistic state estimation, and visual odometry. It uses ROS as middleware and Python exercises. Outside the technical scope, EdX also offers the "Drones and Autonomous Systems" course [37], mainly focused on fundamentals and applications.

Coursera offers the "Aerial Robotics" course [38], provided by the University of Pennsylvania. It introduces 2D and 3D geometry, mechanics of flight in micro aerial vehicles and teaches how to develop dynamic models, derive controllers, and synthesize planners for operating in three-dimensional environments. It uses MATLAB as programming middleware.

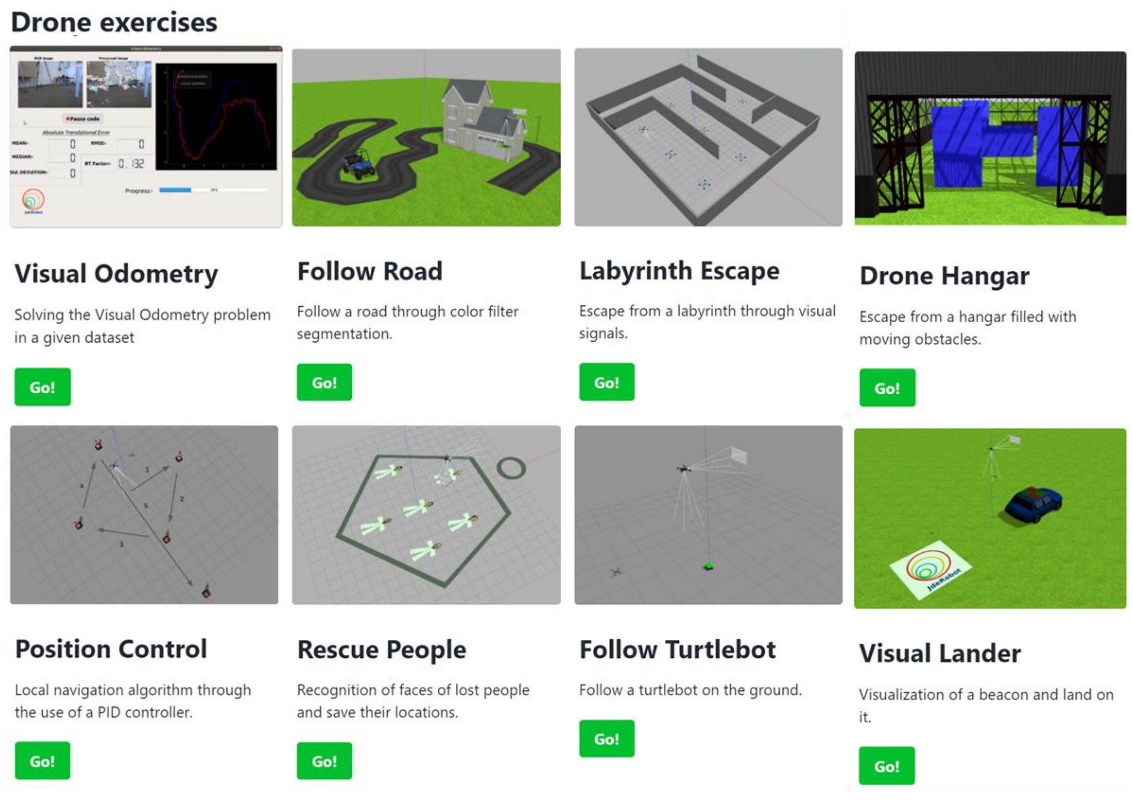
Udacity, a spin-out of Stanford university focused on online education, offers the "Flying Car and Autonomous Flight Engineering" nano degree [39]. It puts emphasis on 3D motion planning, control algorithms, and required estimation techniques such as extended Kalman filters. For instance, it teaches how to fly a drone around a complex urban environment, load a real city map, plan a collision-free path between buildings, and watch the drone fly above city streets. It uses C++ as the programming language.

The Udemy platform offers the "Drone Programming Primer for Software Development" course [40]. It teaches the open-source flight stack based in ArduPilot flight controller, MAVLink, and DroneKit middleware. This course uses the ArduPilot software in the loop (SITL) simulator, which simulates an ArduPilot-based autopilot and communicates with it using MAVLink over the local IP network. The drone is modeled as an object in a Python script, allowing the student to command a real or simulated drone using Python. DroneKit has been used in many drone-based research papers [41,42]. This open-source flight stack transcends its hobbyist roots and is branching out into business applications at a high rate.

Additionally, some drone-based tutorials have begun to appear in some outstanding conferences. For instance, the "Drone Vision and Deep Learning" tutorial [43] by Ioannis Pitas, from the IEEE/CVF Conference on Computer Vision and Pattern Recognition CVPR2020, focuses on advanced computer vision techniques to be run onboard UAVs. They deal with semantic world mapping, detection of target/obstacle/crowd/point of interest, and 2D/3D target tracking. Moreover, they facilitate drone autonomy as drone vision plays a pivotal role in drone perception/control, coping with the limited computing resources available onboard.

3. RoboticsAcademy Learning Environment

RoboticsAcademy (http://jderobot.github.io/RoboticsAcademy/) is an open-access platform containing a collection of self-contained exercises to learn robotics and computer vision practically, oriented to engineering education. The proposed activities (http://jderobot.github.io/RoboticsAcademy/exercises/) cover a wide range of topics, including autonomous cars, mobile robots, industrial robotics, and drones. As an evolution from its initial design [44], RoboticsAcademy is periodically updated with new exercises and functionalities. Some of the exercises provided by the RoboticsAcademy educational platform are depicted in Figure 1.


Drone exercises

Visual Odometry
Solving the Visual Odometry problem in a given dataset

Follow Road
Follow a road through color filter segmentation.

Labyrinth Escape
Escape from a labyrinth through visual signals.

Drone Hangar
Escape from a hangar filled with moving obstacles.

Position Control
Local navigation algorithm through the use of a PID controller.

Rescue People
Recognition of faces of lost people and save their locations.

Follow Turtlebot
Follow a turtlebot on the ground.

Visual Lander
Visualization of a beacon and land on it.

Figure 1. Some of the exercises available in RoboticsAcademy.

In each exercise, the student is challenged with a specific problem, which is expected to be solved by programming standard robotics algorithms (e.g., perception, planning, and control) to provide the robot with the necessary intelligence to complete the assigned task. Students shall program their solutions using the Python programming language, selected due to its gradual learning curve and growing popularity in robotics [45] and other fields.

The platform provides a simple application programming interface (API) to grant high-level access to robot sensors and actuators for each exercise. This simplicity allows students to focus on the algorithms' coding rather than battling with time-consuming hardware management details. From an architectural point of view, each exercise is divided into three different layers (see Figure 2):

1. Hardware layer: represents the robot itself, being a real robot or a simulated one.
2. Middleware layer: includes the software drivers required to read information from robot sensors and control robot actuators.
3. Application layer: contains the student's code (algorithms implemented by the student to solve the problem proposed) and a template (internal platform functions that provide two simple APIs to access robot hardware and take care of graphical issues for students' code debugging purposes).

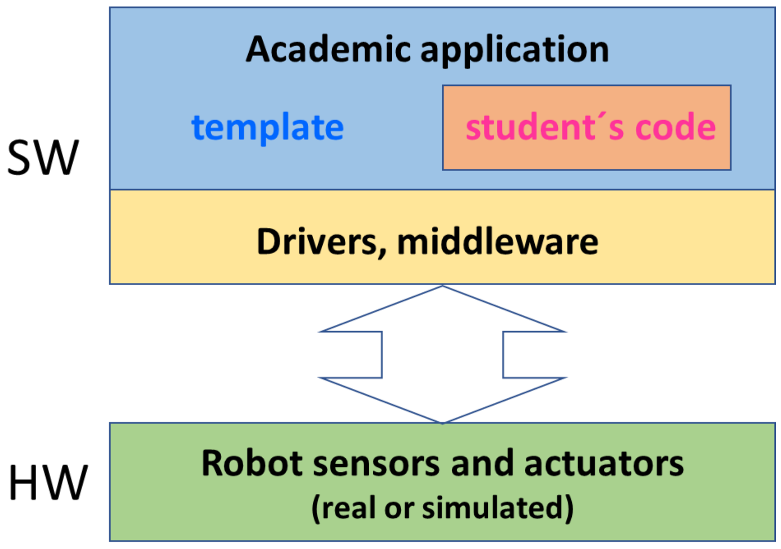


Figure 2. Design of exercises in RoboticsAcademy.

3.1. *Hardware Layer*

As opposed to other robotics teaching platforms [11], the exercises proposed in RoboticsAcademy include a wide variety of robots (drones, mobile robots with wheels, cars, humanoids, industrial manipulators) in real or simulated arrangements. Likewise, practical activities with popular sensors such as lasers, GPS and RGB-D cameras (Kinect, Xtion) can be performed.

The internal design of RoboticsAcademy allows the same student's code to be run in a physical robot (when available) or its simulated counterpart, with only minor configuration changes. For the 3D simulation of the different robots and environments, the open-source Gazebo simulator [46] has been used. The Gazebo simulator is widely recognized among the international robotics community.

Two different flight controllers are conventional when using drones: ArduPilot and PX4. They implement the low-level control and stabilization of the flying robot, leading to flight missions' execution as sequences of waypoints. They run onboard the drones themselves, allowing communication with ground control stations or controlling applications. Regarding the presented course hardware, PX4 is the flight controller supported in RoboticsAcademy, both on real drones and in the Gazebo simulator.

3.2. *Middleware Layer*

ROS [47] is the adopted middleware in RoboticsAcademy. ROS is the de facto standard middleware, with the largest community of users in the robotics field. It provides a comprehensive collection of drivers for a wide variety of robots, helping to shorten the programming development time and increase the robustness of final applications due to code reuse. It supports many programming languages, including Python. The current version used in RoboticsAcademy is ROS-Melodic.

Concerning the middleware needed in the presented course, the MAVLink protocol [48] connects the student's application with the drone flight controller. ROS supports MAVLink through its

MAVROS package. For onboard drone cameras' support, regular ROS image drivers are used. In RoboticsAcademy, MAVROS has been extended to support speed regulation of the flying control. In this way, the drone programming interface in RoboticsAcademy provides both the high-level classic waypoint missions and the mid-level speed commands to rule drone movements. Speed commands allow more fine-grained control of the drone, such as the one required in the flying robots' visual control applications.

3.3. Application Layer

The application layer comprises the student's code and the exercise template, which acts as a container of the student's code and provides the following services:

- Hardware abstraction layer (HAL) API: simple API to obtain processed data from robot sensors and command robot actuators at a high level
- Graphical user interface (GUI) API: used for student's code checking and debugging, shows sensory data or partial processing in a visual way (for example, the images from the robot camera or the results of applying color filtering)
- Timing skeleton for implementing the reactive robot behavior: a continuous loop of iterations, executed several times per second. Each iteration performs four steps: collecting sensory data, processing them, deciding actions, and sending orders to the actuators

The application layer is implemented as a ROS node. The template provided in each node supports access to the robot's sensors and actuators using the appropriate ROS methods and communication mechanisms (topics, subscriptions, publishing . . .). The hardware complexity and details are hidden to grant a simple local API in Python for the student (HAL API), employing ROSpy (a pure Python client library for ROS). Figure 3 shows a typical execution of an exercise using a simulated robot.

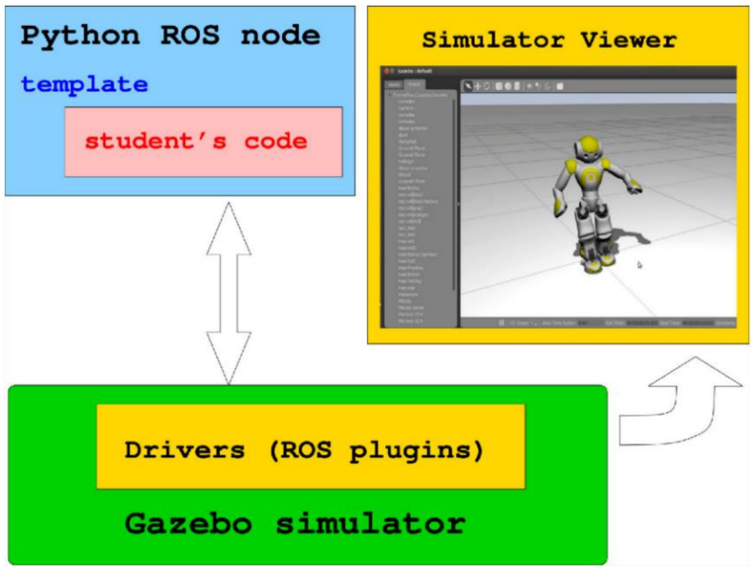


Figure 3. Typical execution of an exercise of RoboticsAcademy using Gazebo viewer.

The drone HAL API used in all the exercises of the presented course includes these methods for getting sensor measurements, images, and drone state:

- drone. get_position (): returns the actual position of the drone as a 1×3 NumPy array [x, y, z], in meters.
- drone. get_velocity (): returns the actual velocities of the drone as a 1×3 NumPy array [vx, vy, vz], in m/s.

- drone. get_yaw_rate (): returns the actual yaw rate of the drone, in rad/s.
- drone. get_orientation (): returns the actual roll, pitch, and yaw of the drone as a 1×3 NumPy array [roll, pitch, yaw], in rad.
- drone. get frontal_image (): returns the latest images from the frontal camera as an OpenCV image (cv2_image).
- drone. get_ventral_image (): returns the latest images from the ventral camera as an OpenCV image (cv2_image).
- drone. get_landed_state (): returns one if the drone is on the ground (landed), two if it is in the air, and four if landing.

Additionally, the HAL API provides these methods for drone control:

- drone.set_cmd_vel (vx, vy, vz, yaw_rate): commands the linear velocity of the drone in the x, y, and z directions (in m/s) and the yaw rate (rad/s) in its body-fixed frame.
- drone.set_cmd_mix (vx, vy, z, yaw_rate): commands the linear velocity of the drone in the x, y directions (in m/s), the height (z) related to the takeoff point, and the yaw rate (in rad/s).
- drone.takeoff (height): takes off from the current location to the given elevation (in meters).
- drone. land (): lands at the current location.

RoboticsAcademy native release is an open-source platform, free and simple to use. All of its underlying infrastructure (Python, Gazebo, ROS) is provided as regular and standard official Debian packages, and so they can be installed easily on Ubuntu Linux computers. Additionally, the required underlying resources, which are not contained in traditional Gazebo or ROS packages, are created and stored in several Debian packages from the JdeRobot organization.

The native release runs on Ubuntu Linux machines. Docker images have been created and are maintained for Windows and macOS users so that they can use the platform through a virtual machine (Docker container) in their computers.

4. Drone Programming Course

An open-access drone programming distance course is proposed in this article, based on the RoboticsAcademy platform. This course is freely available, ready to use by anyone from anyplace with Internet access. It includes both theoretical and practical contents (focusing on the "learning by doing" concept). Although it is recommended to have some previous programming experience to facilitate the realization of practical exercises, it has no relevant prerequisites. These exercises help students to consolidate theoretical concepts, focusing on programming skills for UAVs.

4.1. Course Syllabus

Drone autonomous applications, as ground robot applications, may be seen as the onboard combination of perception and control. The course presents the most common drone sensors and actuators. It also describes position-based and vision-based control techniques, which are the main objectives of the course. Additionally, it introduces map-based global navigation in conjunction with relevant self-localization algorithms, allowing drone 3D navigation even in indoor and GPS-denied environments.

The theoretical content of the course is divided into six different units, as follows:

- Unit 1. Introduction to aerial robotics: types of UAVs, real drone applications such as military, logistics, visual inspection.
- Unit 2. Drone sensors and perception: inertial measurement units (IMUs), compass, GPS, LIDAR, cameras, elementary image processing.
- Unit 3. Flight physics and basic control: 3D geometry, quadrotor physics, basic movements, hovering, forward motion, rotation, stabilization.

- Unit 4. Drone control: reactive systems, proportional–integral–derivative (PID) controllers, fuzzy control, finite state machines, position-based control, vision-based control.
- Unit 5. Drone global navigation: 2D and 3D path planning.
- Unit 6. Visual self-localization: ORBSlam, semi-direct visual odometry (SVO) algorithms.

4.2. Course Practical Exercises

The practical content of the course is divided into several exercises, enumerated in Table 1 below. Two drones are mainly used: the ArDrone (with a frontal camera) and the 3DRobotics Iris (with two cameras, frontal, and ventral). They run on simulation, but the real drones may also be programmed using RoboticsAcademy if they are available for the students.

Table 1. List of exercises available for the course.

Drones	
DR1	Navigation by position
DR2	Following an object on the ground
DR3	Following a road with visual control
DR4	Cat and mouse
DR5	Escape from a maze following visual clues
DR6	Searching for people to rescue within a perimeter
DR7	Escape from a hangar with moving obstacles
DR8	Land on a beacon
Computer Vision	
CV1	Color filter for object detection
CV2	Visual odometry for self-localization

Some of the most representative exercises are detailed below:

- Navigation by position (http://jderobot.github.io/RoboticsAcademy/exercises/Drones/position_control). This exercise aims to implement an autopilot by using the GPS sensor, the IMU, and a position-based PID controller. For this exercise, a simulated 3D world has been designed that contains the quadrotor and five beacons arranged in a cross. The objective is to program the drone to follow a predetermined route visiting the five waypoints in a given sequence, as shown in Figure 4. It illustrates the algorithms typically included in commercial autopilots such as ArduPilot or PX4.
- Following an object on the ground (http://jderobot.github.io/RoboticsAcademy/exercises/Drones/follow_turtlebot). In this exercise, the objective is to implement the logic that allows a quadrotor to follow a moving object on the ground (an autonomous robot named Turtlebot with a green-colored top, see Figure 5), using a primary color filter in the images and a vision-based PID controller. The drone keeps its altitude and moves only in a 2D plane.
- Landing on a moving car (http://jderobot.github.io/RoboticsAcademy/exercises/Drones/visual_lander). In this exercise, the student needs to combine pattern recognition and vision-based control to land on a predefined beacon, a four-square chess pattern on the roof of a moving car (see Figure 6). The required image processing is slightly more complicated than a simple color filter, as the beacon may be partially seen, and its center is the relevant feature. Likewise, the controller needs to command the vertical movement of the drone.
- Escape from a maze using visual clues (http://jderobot.github.io/RoboticsAcademy/exercises/Drones/labyrinth_escape). In this exercise, the student needs to combine local navigation and computer vision algorithms to escape from a labyrinth with the aid of visual clues. The clues are green arrows placed on the ground, indicating the direction to be followed (see Figure 7). Pattern recognition in real-time is the focus here, as fast detection is essential for the drone.

- Searching for people to rescue within a perimeter (http://jderobot.github.io/RoboticsAcademy/exercises/Drones/rescue_people). The objective of this exercise is to implement the logic of a global navigation algorithm to sweep a specific area systematically and efficiently (foraging algorithm), in conjunction with visual face-recognition techniques, to report the location of people for subsequent rescue (Figure 8). The drone behavior is typically implemented as a finite state machine, with several states such as go-to-the-perimeter, explore-inside-the-perimeter, or go-back-home.

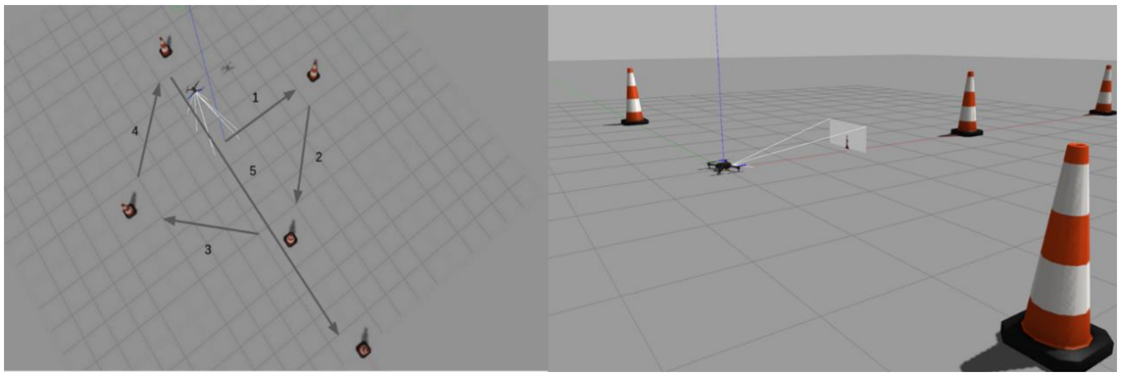


Figure 4. "Navigation by position" exercise.

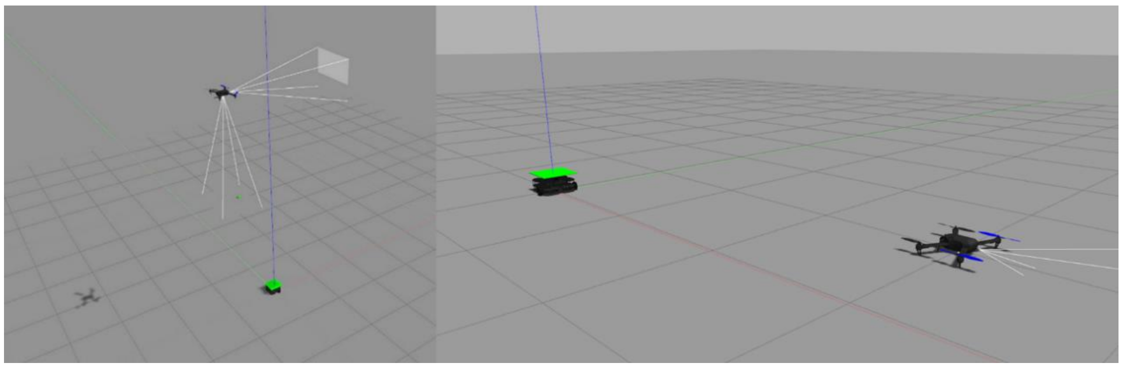


Figure 5. "Following an object on the ground" exercise.

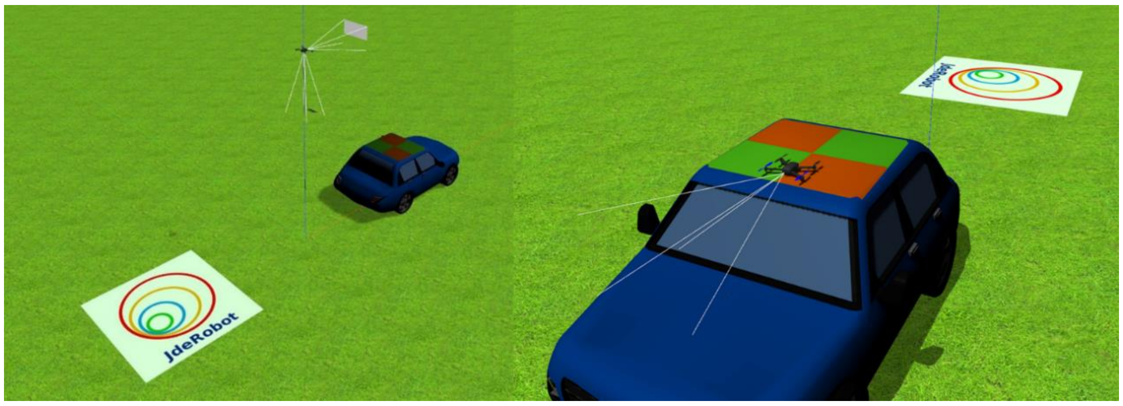


Figure 6. "Landing on a moving car" exercise.

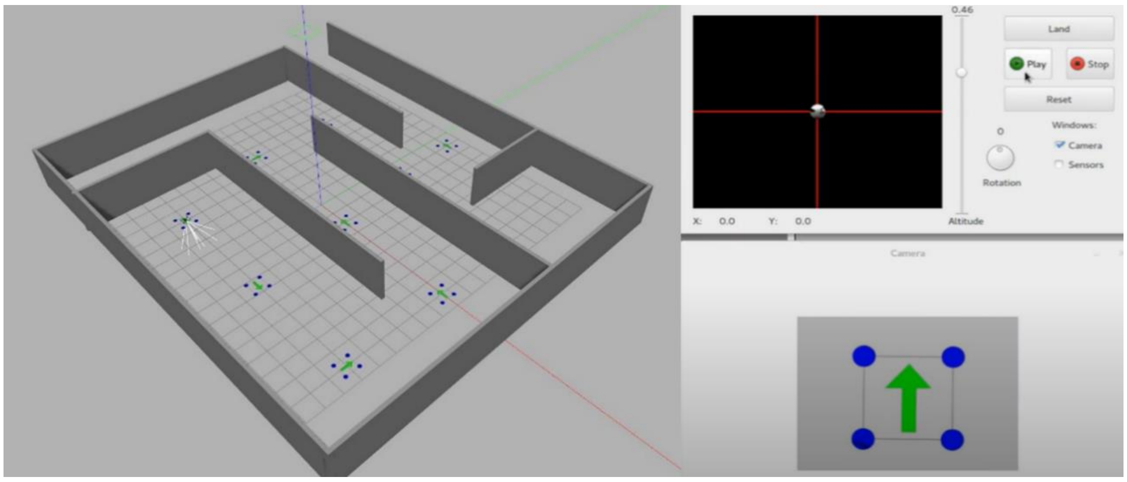
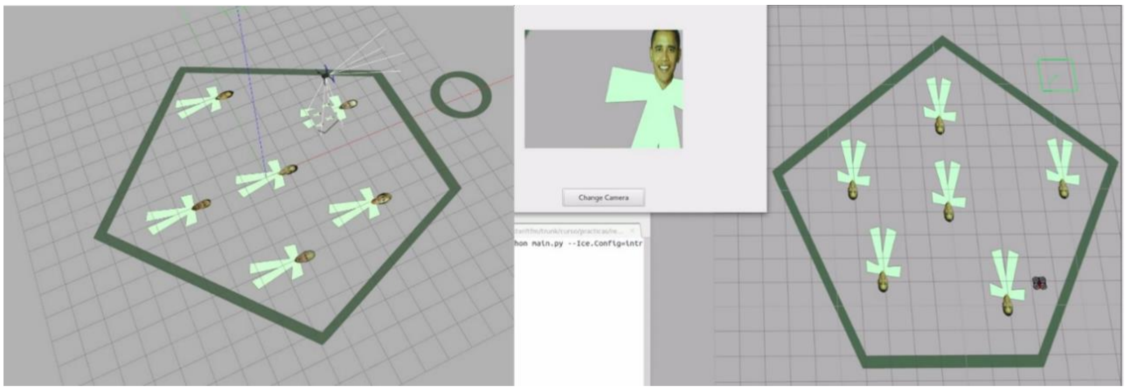


Figure 7. "Escape from a maze using visual clues" exercise.



```
hon main.py --Ice.Config=intr
```

Figure 8. "Searching for people to rescue within a perimeter" exercise.

4.3. Lessons Learned

The course proposed has already been taught successfully with students from the Rey Juan Carlos University, specifically from several engineering degrees, testing the correct platform functioning and validating the exercises' content over five academic years. The course syllabus covers the usual agendas of engineering degrees, also fitting into the European higher educational framework.

The course's objective is to introduce students to state of the art drone programming: the essential components of drones (sensors and actuators), the fundamental problems of drone programming (navigation, flight control, localization, etc.), and the most successful techniques and algorithms for addressing them. The drones' mechanical and electronic aspects have been considered, with particular emphasis placed on their programming algorithms; in this aspect resides their intelligence (for example, architectures of reactive control to generate autonomous behavior). Factors related to computer vision in drones (based on artificial intelligence), a growing trend in drone applications, are also included.

Our impression about the utilization of RoboticsAcademy on this course is that it was very motivating for the students, taking as reference the same subject taught during the 2017/2018 edition when RoboticsAcademy was not yet used. Most comments extracted from student surveys indicate that the students remarkably welcomed utilizing a platform that facilitates distance drone programming. For instance, Figure 9 shows their evaluation of the practice environment in the 2018/2019 course in a survey with 21 students, yielding that 57.1% of the students had an excellent opinion of it, and 42.9% of them were more than satisfied.

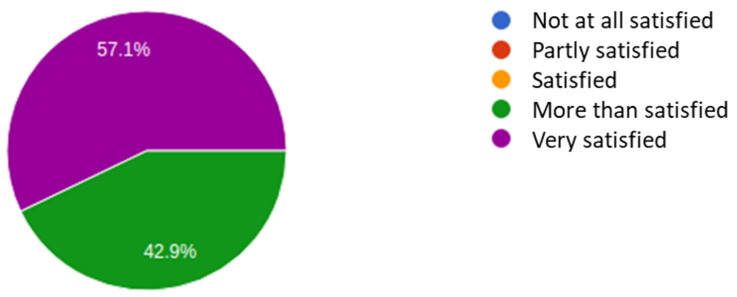


Figure 9. Results of the survey for the 2018/2019 course.

As a result of the lessons learned during the first editions of the course, both the platform and the course itself mostly evolved, currently integrating the following benefits:

- Easy and time-efficient installation, allowing the students to start programming drones earlier: installation previously consumed at least two weeks to prepare the programming environment, as it was based on Linux packages and virtual machines. With the current framework release, installation recipes use binary Debian packages (for students with Linux operating system) and Docker containers (for students with Windows or macOS). These Docker containers have the environment already pre-installed, being installed themselves in a fast way. In the 2018/2019 course, 85.7% of students used RoboticsAcademy in native Linux, while 14.3% used Microsoft Windows with Docker containers (Figure 10). As reported in the surveys, the user experience was pleasant on both systems.
- The same platform is used for programming simulated and real drones: previously, students had to learn different tools to program different drones (both simulated and real), resulting in less time to learn programming and algorithms.
- Distance learning: the course was designed to be taught remotely; therefore, any student could learn and practice from home, anytime.
- The forum proved to be a precious tool for intercommunication and doubt resolution.

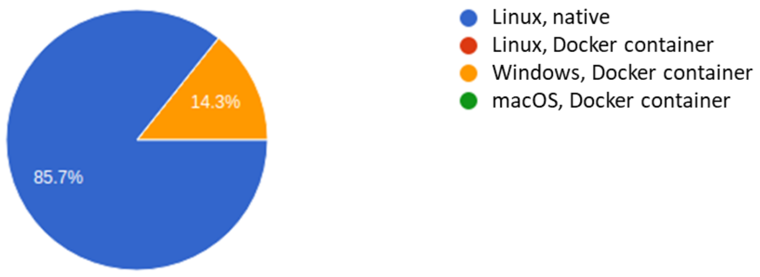


Figure 10. Results of the survey for the 2018/2019 course.

All the details of MAVROS, MAVLink, and PX4 have been hidden from students that use the simple drone HAL API described in Section 3.3 for drone programming. In this way, students may focus on navigation and control algorithms, which are the course's core objective. Therefore, this practice environment is not suitable for learning the tools mentioned above, which can also be useful for drone engineering.

5. Drone Programming Competitions

Gamification is the term commonly used to refer to game-based learning systems. In these systems, specific aspects found in games, such as making competitions, establishing rankings and scoreboards, and giving rewards or badges, are implemented in the classroom to make the learning process more attractive to the student [49].

It is already known that gamified learning in higher education (and in engineering degrees in particular) enhances students' engagement and motivation and improves their attendance ratios and grades [50,51]. Positive reinforcement in both on-site and distance classrooms is promoted when managing the student load using challenges of progressive difficulty and gamified step-by-step tasks [52]. Additionally, gamification provides instant student feedback and grading, proven to help formative self-assessment and auto evaluation [53].

Through stepwise updates, RoboticsAcademy has been redesigned to support gamified learning in computer vision and robotics, including:

- Competitive exercises. Several exercises in our present collection are based on completing a task in a given time while competing against the fastest or more complete solution, increasing the student's engagement.
- Social interactions between students, teachers, and developers promoted utilizing a dedicated RoboticsAcademy (https://developers.unibotics.org/) web forum and several accounts across prime social media platforms since 2015, such as a video channel on YouTube (https://www.youtube.com/channel/UCgmUgpircYAv_QhLQziHJOQ) with more than 300 videos and 40,000 single visits and a Twitter account with more than 200 tweets (https://twitter.com/jderobot).

Beyond being used in university courses, as explained in Section 4.3, RoboticsAcademy and its drone exercises have also been used as a convenient tool for organizing both on-site and distance competitions in Robotics, particularly within our Program-A-Robot event series. This championship was born in 2016 as a competition to foster robotics and computer vision, proposing challenging problems to the participants and offering the possibility to measure and automatically rank your solution with other fellow competitors.

The targets in the first edition were undergrad engineering students at Rey Juan Carlos University (https://jderobot.github.io/activities/competitions/2016). The second edition was organized as a national event inside the Spanish Jornadas Nacionales de Robotica (https://jderobot.github.io/activities/competitions/2017) in 2017. The third event was a distance international competition aimed at undergraduate, graduate, or Ph.D. engineering students worldwide. Its final round was held as part of the International Conference on Intelligent Robots and Systems IROS 2018 (https://jderobot.github.io/activities/competitions/2018), which took place in Madrid in October 2018, and was live-streamed on YouTube.

Two different challenges related to programming the autonomous intelligence of vision-assisted mobile aerial robots were proposed, focusing on this last competition.

In the first exercise, the cat-and-mouse challenge, a quadcopter aerial robot called "cat drone" had to be programmed to search, chase, and stay at a given distance to another aerial quadcopter, the "mouse drone" (see Figure 11).

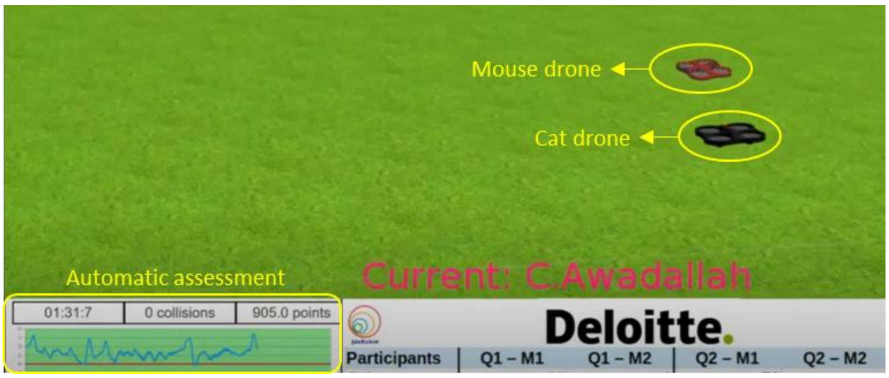


Figure 11. "Cat and mouse" exercise.

The organizing committee programmed several different autonomous mice drones to allow for three qualifying sessions, starting from more straightforward movement behaviors, and transitioning to more complex behaviors to track mice. Some of them were released for training. An automatic assessment tool (referee) was also developed to score each 2 min round systematically. The referee measured the instantaneous distance from each cat drone to the mouse, showing its evolution (along with the whole game) on the screen. When it was below a certain threshold (close-threshold), both the progress line and the background were painted green, and when it was above in red. As long as the cat drone was near the mouse, the score increased. It was forbidden to touch the mouse, with each contact penalized in the final round score.

In the second challenge, named escape from the hangar, a single mobile aerial robot had to take off inside the hangar and find its way out in less than 1 min and avoiding the collision with six inner walls, which were moving in a random pattern to increase the difficulty (see Figure 12). Again, an automatic referee was built to measure how many seconds each aerial robot takes to leave the hangar. The initial score started from 60 points and decreased during the time each robot was inside the hangar. It was forbidden to touch the walls, and each contact further reduced the score. Additionally, the drone's starting point was changed during the competition, so a pure position-based control was not a proper strategy.

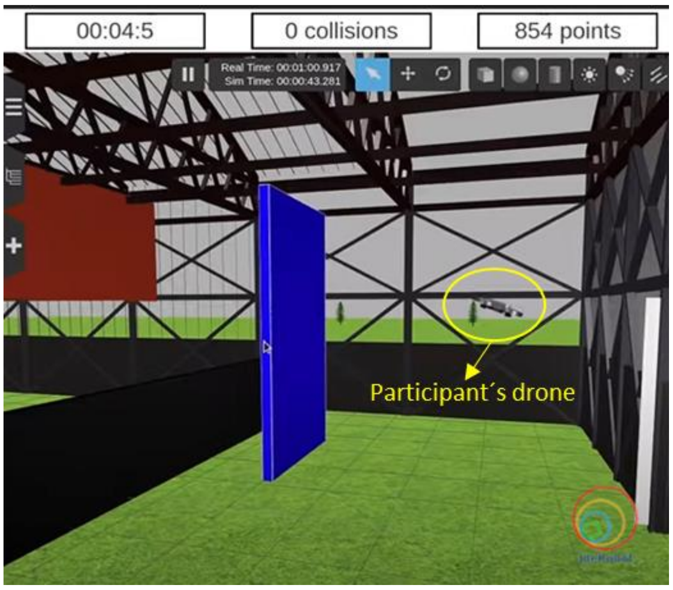


Figure 12. "Escape from hangar" exercise.

The video of the final round in the IROS2018-Program-A-Robot competition is available online (https://www.youtube.com/watch?v=_0ZkciOHnmU). It shows the performance of the uploaded solutions for the best three participants in each challenge, together with their automatic evaluations for each 2 min rounds. In summary, it was an excellent acid test for the capability of RoboticsAcademy as an online platform for holding, among others, distance competitions in robotics and computer vision.

More than 40 students participated in the competitions (Figure 13). The feedback from them was very positive in the three editions, which encouraged us to scale up in scope from a local competition to a national one, and finally to an international one. For instance, "Thank you for organizing this drone programming competition; this type of enterprise is really cool" from one participant from the 2017 edition is an illustrative example. Another participant of IROS2018 edition even kept engaged with RoboticsAcademy after the competition and developed a new exercise for it, the "Visual Odometry with RGBD Camera" exercise (https://jderobot.github.io/RoboticsAcademy/exercises/ComputerVision/visual_odometry) as an open-source contribution to the project.

Figure 13. Participants in 2016 and 2017 editions of Program-A-Robot competition.

6. Summary and Conclusions

The course presented in this article is a free educational tool for teaching drone programming in higher education in engineering, particularly suitable for distance learning. It comprises several independent exercises that mimic drones' real-life practical applications, including visual landing, search-and-rescue missions, and road following in real or simulated arrangements. These exercises focus on perception, planning-and-control algorithms, emphasizing state-of-the-art topics as computational vision or artificial intelligence. All activities are proposed to the students with a practical "learning by doing" approach.

Regarding the RoboticsAcademy platform, its main features are, in summary:

- Each exercise is divided into three layers: hardware, middleware, and application. This internal design makes it easier to run the same student's code in physical and simulated robots with only minor configuration changes. The open-source Gazebo simulator, widely recognized in the robotics community, has been used for 3D simulations of all robots, sensors, and environments.
- The middleware layer is ROS-based, the de facto standard in service robotics that supports many programming languages, including Python. Previously, RoboticsAcademy was separated from this widely used middleware, needing frequent maintenance, and having little scalability and a short number of users (only those who already had a thorough background with the tool).
- The application layer contains the student's code. It includes a hardware abstraction layer (HAL) API to grant students high-level access to the robot sensors and actuators, and a graphical user interface (GUI) API to show sensor data, process images, or debug code.
- RoboticsAcademy runs natively on Linux Ubuntu machines and can be installed using official Debian packages. Docker containers are used to allow easy installation in Windows and macOS based computers, making RoboticsAcademy particularly suited to the distance learning approach.

An open-access distance full course covering diverse drone programming aspects has been proposed, aimed at engineering students. Its syllabus comprises six academic units, comprising from sensors and actuators to navigation and computer vision. The course includes practical content, all covered with RoboticsAcademy exercises. It was successfully tested and validated with students of several engineering degrees at the Rey Juan Carlos University over five academic years. In this course, any student could learn and practice from home, anytime. End-of-course surveys showed that using the platform was helpful, productive, and motivating for them.

RoboticsAcademy also includes additional functionalities as automatic assessment tools and competitive goals, making it a perfectly adapted tool for gamified learning strategies. It has been proven as a convenient platform for organizing online robotics competitions as well. The Program-A-Robot annual championship was born in 2016. It has already celebrated its third edition using RoboticsAcademy, proposing competitive challenges about programming the autonomous intelligence of different vision-assisted mobile robots.

Future work lines for enhancing RoboticsAcademy include developing more exercises, expanding the present full course, and proposing different specific learning paths to reach a broader range of educational stages. Furthermore, an in-depth experimental analysis is planned to provide quantitative evidence about user experience, creating a complete questionnaire, and collecting information from other students. It will provide a more solid methodological basis.

Author Contributions: Conceptualization, J.M.C. and D.M.-M.; formal analysis, J.M.C.; funding acquisition, J.M.C.; investigation, P.A. and D.R.-Á.; methodology, P.A.; project administration, J.M.C. and J.V.; resources, J.V.; software, D.M.-M., P.A., D.R.-Á. and J.F.-C.; supervision, J.M.C.; validation, D.M.-M. and L.G.-P.; visualization, L.G.-P.; writing—original draft, J.V., L.G.-P. and J.F.-C.; writing—review and editing, J.M.C. and J.F.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the Community of Madrid in the framework of two research projects: (1) Multiannual agreement with Rey Juan Carlos University in line of action 1, "Encouragement of Young Ph.D. Students Investigation" project ref. F664, acronym UNIBOTICS2.0. (2) RoboCity2030-DIH-CM (2019-2022): RoboCity2030—Madrid Robotics Digital Innovation Hub, Programa de Actividades de I+D entre Grupos de investigación de la Comunidad de Madrid en Tecnologías 2018. Project ref. S2018/NMT-4331.

Acknowledgments: The authors thank Google for funding the JdeRobot non-profit organization in its calls for Google Summer of Code 2015, 2017, 2018, 2019, and 2020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Aliane, N. Teaching fundamentals of robotics to computer scientists. *Comput. Appl. Eng. Educ.* **2011**, *19*, 615–620. [CrossRef]
2. Mateo, T.; Andujar, J. Simulation tool for teaching and learning 3d kinematics workspaces of serial robotic arms with up to 5-DOF. *Comput. Appl. Eng. Educ.* **2012**, *20*, 750–761. [CrossRef]
3. Mateo, T.; Andujar, J. 3D-RAS: A new educational simulation tool for kinematics analysis of anthropomorphic robotic arms. *Int. J. Eng. Educ.* **2011**, *27*, 225–237.
4. Lopez-Nicolas, G.; Romeo, A.; Guerrero, J. Simulation tools for active learning in robot control and programming. In Proceedings of the 20th EAEEIE Annual Conference, Valencia, Spain, 22–24 June 2009; Innovation in Education for Electrical and Information Engineering: New York, NY, USA, 2009.
5. Lopez-Nicolas, G.; Romeo, A.; Guerrero, J. Active learning in robotics based on simulation tools. *Comput. Appl. Eng. Educ.* **2014**, *22*, 509–515. [CrossRef]
6. Jara, C.; Candelas, F.; Pomares, J.; Torres, F. Java software platform for the development of advanced robotic virtual laboratories. *Comput. Appl. Eng. Educ.* **2013**, *21*, 14–30. [CrossRef]
7. Gil, A.; Reinoso, O.; Marin, J.; Paya, L.; Ruiz, J. Development and deployment of a new robotics toolbox for education. *Comput. Appl. Eng. Educ.* **2015**, *23*, 443–454. [CrossRef]
8. Fabregas, E.; Farias, G.; Dormido-Canto, S.; Guinaldo, M.; Sanchez, J.; Dormido, S. Platform for teaching mobile robotics. *J. Intell. Robot. Syst.* **2016**, *81*, 131–143. [CrossRef]
9. Detry, R.; Corke, P.; Freese, M. TRS: An Open-Source Recipe for Teaching/Learning Robotics with a Simulator. 2014. Available online: http://ulgrobotics.github.io/trs (accessed on 16 December 2020).
10. Guzman, J.; Berenguel, M.; Rodriguez, F.; Dormido, S. An interactive tool for mobile robot motion planning. *Robot. Auton. Syst.* **2008**, *56*, 396–409. [CrossRef]
11. Guyot, L.; Heiniger, N.; Michel, O.; Rohrer, F. Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions. In Proceedings of the 2nd International Conference on Robotics in Education (RiE 2011), Vienna, Austria, 15–16 September 2011; Stelzer, R., Jafarmadar, K., Eds.; Ghent University: Ghent, Belgium, 2011; pp. 53–58.
12. Soto, A.; Espinace, P.; Mitnik, R. A mobile robotics course for undergraduate students in computer science. In Proceedings of the 2006 IEEE 3rd Latin American Robotics Symposium (LARS'06), Santiago, Chile, 26–27 October 2006; IEEE: New York, NY, USA, 2007; pp. 187–192.
13. Thrun, S. Teaching challenge. *IEEE Robot. Autom. Mag.* **2006**, *13*, 12–14. [CrossRef]
14. Jara, C.A.; Candelas, F.A.; Puente, S.; Torres, F. Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. *Comput. Educ.* **2011**, *57*, 2451–2461. [CrossRef]

15. Cliburn, D.C. Experiences with the LEGO Mindstorms throughout the Undergraduate Computer Science Curriculum. In *Frontiers in Education, Proceedings of the 36th Annual Conference, San Diego, CA, USA, 27–31 October 2006*; IEEE: New York, NY, USA, 2007; pp. 1–6. [CrossRef]
16. Gomez-de-Gabriel, J.M.; Mandow, A.; Fernandez-Lozano, J.; García-Cerezo, A.J. Using LEGO NXT Mobile Robots with LabVIEW for Undergraduate Courses on Mechatronics. *IEEE Trans. Educ.* **2011**, *54*, 41–47. [CrossRef]
17. Cuéllar, M.; Pegalajar Jiménez, M. Design and Implementation of Intelligent Systems with LEGO Mindstorms for Undergraduate Computer Engineers. *Comput. Appl. Eng. Educ.* **2014**, *22*, 153–166. [CrossRef]
18. Montés, N.; Rosillo, N.; Mora, M.C.; Hilario, L. Real-Time Matlab-Simulink-Lego EV3 Framework for Teaching Robotics Subjects. In *Proceedings of the International Conference on Robotics and Education RiE 2017*; Springer: Cham, Switzerland, 2018. [CrossRef]
19. Gonzalez-Garcia, S.; Rodríguez, J.; Loreto, G.; Montaño Serrano, V. Teaching forward kinematics in a robotics course using simulations: Transfer to a real-world context using LEGO mindstorms™. *Int. J. Interact. Des. Manuf.* **2020**, *14*. [CrossRef]
20. Zhang, M.; Wan, Y. Improving Learning Experiences Using LEGO Mindstorms EV3 Robots in Control Systems Course. *Int. J. Electr. Eng. Educ.* **2020**. [CrossRef]
21. Esposito, J.M. The state of robotics education: Proposed goals for positively transforming robotics education at postsecondary institutions. *IEEE Robot. Autom. Mag.* **2017**, *24*, 157–164. [CrossRef]
22. Corke, P.; Greener, E.; Philip, R. An Innovative Educational Change: Massive Open Online Courses in Robotics and Robotic Vision. *IEEE Robot. Autom. Mag.* **2016**, *23*, 81–89. [CrossRef]
23. Artificial Intelligence for Robotics. Available online: https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373 (accessed on 5 November 2020).
24. Autonomous Mobile Robots. Available online: https://www.edx.org/course/autonomous-mobile-robots-ethx-amrx-1 (accessed on 5 November 2020).
25. Pozzi, M.; Malvezzi, M.; Prattichizzo, D. Mooc on the art of grasping and manipulation in robotics: Design choices and lessons learned. In Proceedings of the International Conference on Robotics and Education RiE 2017, Sofia, Bulgaria, 26–28 April 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 71–78.
26. Kulich, M.; Chudoba, J.; Kosnar, K.; Krajnik, T.; Faigl, J.; Preucil, L. Syrotek-distance teaching of mobile robotics. *IEEE Trans. Educ.* **2013**, *56*, 18–23. [CrossRef]
27. Zalewski, J.; Gonzalez, F. Evolution in the Education of Software Engineers: Online Course on Cyberphysical Systems with Remote Access to Robotic Devices. *Int. J. Online Eng.* **2017**, *13*, 133–146. [CrossRef]
28. Téllez, R.; Ezquerro, A.; Rodríguez, M.Á. *ROS in 5 Days: Entirely Practical Robot Operating System Training*; Independently Published: Madrid, Spain, 2016.
29. Casañ, G.A.; Cervera, E.; Moughlbay, A.A.; Alemany, J.; Martinet, P. ROS-based online robot programming for remote education and training. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; IEEE: New York, NY, USA, 2015; pp. 6101–6106.
30. Cervera, E.; Martinet, P.; Marin, R.; Moughlbay, A.A.; Del Pobil, A.P.; Alemany, J.R.; Casañ, G. The robot programming network. *J. Intell. Robot. Syst.* **2016**, *81*, 77–95. [CrossRef]
31. Casañ, G.; Cervera, E. The Experience of the Robot Programming Network Initiative. *J. Robot.* **2018**, *2018*, 2312984. [CrossRef]
32. Cervera, E.; Del Pobil, A.P. Roslab: Sharing ROS Code Interactively with Docker and Jupyterlab. *IEEE Robot. Autom. Mag.* **2019**, *26*, 64–69. [CrossRef]
33. Liu, Y.; Xu, Y. Summary of cloud robot research. In Proceedings of the 2019 25th International Conference on Automation and Computing (ICAC), Lancaster, UK, 5–7 September 2019; IEEE: New York, NY, USA, 2019; pp. 1–5.
34. Autonomous Navigation for Flying Robots. Available online: https://www.edx.org/course/autonomous-navigation-for-flying-robots (accessed on 5 November 2020).
35. Engel, J.; Sturm, J.; Cremers, D. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robot. Auton. Syst.* **2014**, *62*, 1646–1656. [CrossRef]
36. Engel, J.; Sturm, J.; Cremers, D. Camera-based navigation of a low-cost quadrocopter. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Algarve, Portugal, 7–12 October 2012; IEEE: New York, NY, USA, 2012; pp. 2815–2821.
37. Learn the Importance of Autonomous Systems and Drone Technologies. Available online: https://www.edx.org/professional-certificate/umgc-usmx-drones-and-autonomous-systems (accessed on 5 November 2020).

38. Robotics: Aerial Robotics. Available online: https://www.coursera.org/learn/robotics-flight (accessed on 16 December 2020).
39. Flying Car and Autonomous Flight Engineer. Available online: https://www.udacity.com/course/flying-car-nanodegree--nd787 (accessed on 5 November 2020).
40. Drone Programming Primer for Software Development. Available online: https://www.udemy.com/course/drone-programming-primer-for-software-development/ (accessed on 5 November 2020).
41. Psirofonia, P.; Samaritakis, V.; Eliopoulos, P.; Potamitis, I. Use of unmanned aerial vehicles for agricultural applications with emphasis on crop protection: Three novel case-studies. *Int. J. Agric. Sci. Technol.* **2017**, *5*, 30–39. [CrossRef]
42. Freimuth, H.; Müller, J.; König, M. Simulating and executing UAV-assisted inspections on construction sites. In Proceedings of the 34th International Symposium on Automation and Robotics in Construction (ISARC 2017), Taipei, Taiwan, 28 June–1 July 2017; Tribun EU: Brno, Czech Republic, 2017; pp. 647–654.
43. Deep Learning and Multiple Drone Vision. Available online: https://icarus.csd.auth.gr/cvpr2020-tutorial-deep-learning-and-multiple-drone-vision/ (accessed on 5 November 2020).
44. Cañas, J.; Martin, L.J. Innovating in robotics education with gazebo simulator and jderobot framework. In Proceedings of the XXII Congreso Universitario de Innovación Educativa en Enseñanzas Técnicas CUIEET, Alcoy, Spain, 17–19 June 2014; pp. 1483–1496.
45. Joseph, L. *Learning Robotics Using Python*; Packt Publishing: Birmingham, UK, 2015.
46. Koenig, N.P.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, 28 September–2 October 2004; IEEE: New York, NY, USA, 2005; pp. 2149–2154.
47. Quigley, M.; Gerkey, B.; Smart, W.D. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*; O'Reilly Media, Inc.: Newton, MS, USA, 2015.
48. Koubaa, A.; Allouch, A.; Alajlan, M.; Javed, Y.; Belghith, A.; Khalgui, M. Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access* **2019**, *7*, 87658–87680. [CrossRef]
49. Subhash, S.; Cudney, E.A. Gamified learning in higher education: A systematic review of the literature. *Comput. Hum. Behav.* **2018**, *87*, 192–206. [CrossRef]
50. Barata, G.; Gama, S.; Pires Jorge, J.A.; Gonçalves, D. Engaging engineering students with gamification: An empirical study. In Proceedings of the 2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), Poole, UK, 11–13 September 2013; IEEE: New York, NY, USA, 2013.
51. Sanchez-Carmona, A.; Robles, S.; Pons, J. A gamification experience to improve engineering students' performance through motivation. *J. Technol. Sci. Educ.* **2017**, *7*, 150–161. [CrossRef]
52. Reiners, T.; Wood, L. *Gamification in Education and Business*; Springer: Berlin/Heidelberg, Germany, 2015.
53. Menezes, C.; Bortolli, R. Potential of gamification as assessment tool. *Creat. Educ.* **2016**, *7*, 561–566. [CrossRef]