A Short Curriculum on Robotics with Hands-On Experiments in Classroom Using Low-Cost Drones

Sylvain Bertrand, Chiraz Trabelsi, Lionel Prevost

A Short Curriculum on Robotics with Hands-on Experiments in Classroom using Low-Cost Drones

Sylvain Bertrand[1], Chiraz Trabelsi[2], and Lionel Prevost[2]

[1] Université Paris-Saclay, ONERA, Traitement de l'Information et Systèmes
91123, Palaiseau, France
sylvain.bertrand@onera.fr
[2] Learning, Data and Robotics Lab, ESIEA, Paris, France
{chiraz.trabelsi,lionel.prevost}@esiea.fr

Abstract. This paper presents some feedback on a curriculum on robotics developed for Master's level students at ESIEA, a graduate school of engineering in France. The main particularity of this curriculum is that it is composed of only three short modules (control and estimation, computer vision, ROS), with a small number of hours (18hrs each), but proposes hands-on experiments with drones to students. Experiments are done in the classrooms with low-cost drones as exercises integrated into the practical work sessions. The detail of the curriculum, the pedagogic approach, and examples of experiments proposed to the students are presented in this paper.

Keywords: Robotics curriculum · Experiments in classrooms · Low-cost drones · ROS.

1 Introduction

Drones are very motivating platforms for students. They are also of huge interest to teachers from a pedagogical point of view and are now widely used for teaching Science, Technology, Engineering, and Mathematics (STEM), at different levels, ranging from young kids to graduate students [1]. More and more academic curricula devoted to robotics now integrate one or several modules focusing on drones [2]. Similarly, more and more projects realized by students during their curriculum are also oriented to drone design or applications. It is also worth noticing that open resources for education on drones are now easily accessible, see eg. [3][4].

As low-cost hardware and off-the-shelf components or platforms are now available and affordable, practical work and experiments with drones usually complete more theoretical lectures. Different types of hands-on experiments are usually proposed to students during academic modules. The first approach consists in making students develop a drone by designing and/or integrating parts. Most of the time, this type of approach is related to Project-Based Learning,

with hours dedicated to practical work to build a *home-made* drone [5]. Another approach consists in providing students with a representative mockup (eg. a 2-DOF helicopter bench [6]) or an already existing and ready-to-fly vehicle. Some existing platforms that are commonly used for educational purposes are Parrot mini drones, Crazyflie, ARDrone, Beebop, or DJI Tello Edu. In this case, work of the students focuses on more specific technical topics, most of the time designing and implementing algorithms.

From a practical point of view, hands-on experiments with drones by students are usually performed in flight arenas, equipped with nets for safety purposes. When possible, motion capture systems are used for precise indoor localization [2]. Nevertheless, such systems are expensive, and disposing of free space large enough for a flight arena is not always possible in teaching facilities.

In this paper, feedback is proposed on a short curriculum on robotics at ESIEA, a graduate school of engineering in France, which aims at integrating hands-on experiments by students with drones in the classrooms.

The curriculum is dedicated to last-year students, at the Master's level. This is a short curriculum, only composed of three modules, 18 hours each. A strong effort is put into practice since practical work sessions represent half of the hours for two of the modules and almost all the hours for the third one. This is a challenge since the curriculum is also open to students from different backgrounds and has been designed to require very few prerequisites except standard scientific and engineering backgrounds. Another challenge is that the third module also integrates some hands-on experiments with drones as exercises during practical work sessions. Experiments are done by the students in the classroom with low-cost drones and equipment. A link with student projects should also be made in the sense that some students are already working on drone projects at the beginning of this curriculum.

The paper is organized as follows. The next section introduces the context of this curriculum. Its content is then detailed in Section 3. Sections 4 and 5 respectively present the approach proposed to students to progress step-by-step in simulation first and then easily perform hands-on experiments. Examples of such experiments given as exercises to students are also described in Section 5. Before concluding remarks, Section 6 provides some feedback on student projects on drones and their links to this curriculum.

2 Context

The engineering curriculum at ESIEA is composed of three years. The objective of the first one is to teach students all the required scientific and technical background (mathematics, physics, computer science, etc.). The next two years, which correspond to Master's classes, are devoted to more specific curricula, chosen by the students themselves to develop their own expertise in some specific area. Some examples of these *technical majors* are embedded and autonomous systems, software engineering, cyber-security, virtual reality, AI & data science. During the last third year, students can personalize further their experience by

choosing an additional technical or managerial short curriculum (*minor*), eg. robotics, low tech, innovation and entrepreneurship, economic intelligence, etc. Each minor is composed of three modules of 18 hours each and some conferences on additional topics, organized with professionals from the industry.

During the third year, a *technical minor* is devoted to robotics. It is composed of three modules of 18 hours each: control and estimation for mobile robotics, perception and computer vision, and computer programming for robotics (ROS - Robot Operating System). It is open to students from several technical majors with very few specific requisites regarding robotics, automatic control, etc. The syllabus of this *minor* is detailed in the next section.

3 Content of the curriculum

The first module is *Control and estimation for mobile robotics*. It consists of 9 hours of lectures and 9 hours of practical work sessions. The lectures cover the basis of dynamic modeling for mobile robots, pathfinding and trajectory generation, motion control and obstacle avoidance, state estimation, and information fusion for localization. Different fundamental algorithms such as Dijkstra, A*, PID control, potential fields, odometry, inertial navigation, Kalman Filter, etc. are presented to the students with applications to ground mobile robots (differential drive) and drones (quadrotors). Simulation results and videos of real experiments are presented to illustrate the algorithms. Some live demonstrations with real hardware equipment are also proposed during the lectures, mostly concerning sensor technologies (LIDAR, IMU, stereovision).

Three practical work sessions of 3 hours each consist of direct applications of some of the algorithms presented during the lectures: pathfinding, position control with waypoint navigation and obstacle avoidance for a differential drive robot, and position estimation using Kalman filtering. Python codes that implement the algorithms are given to the students except for some parts to be completed during the sessions. When possible codes developed during different parts of the tutorials (eg. path finding and waypoint navigation) are mixed together to enable addressing more complex robotic missions.

The second module is dedicated to *Perception and computer vision for mobile robotics*. It also consists of 9 hours of lectures and 9 hours of practical work sessions. The lectures address the topics of image processing and geometric computer vision, visual odometry, Simultaneous Localization and Mapping (SLAM), environment modeling and cartography, and Machine Learning for computer vision. Students can discover different algorithms and approaches such as features extraction in computer vision (blobs, Harris, SIFT), optical flow (Lucas-Kanade), camera models and calibration, stereovision, point clouds processing, Octomaps, 3D meshes, supervised learning, SVM, neural networks (RNN, CNN), etc. Three tutorial sessions of 3 hours each are dedicated to applying computer vision methods (features extraction, matching, and visual odometry from stereo vision) using Python and OpenCV library.

The third module *computer programming for robotics (ROS)* has been designed in a different way to promote practice as much as possible. The objective is to present the basis and usage of the Robot Operating System by introducing during a short 1.5 hours lecture its basic notions (workspace, ROS master, nodes, topics, messages, services, etc.) and useful tools (rosbag, RViz, RQT, Gazebo simulator). Some examples are presented to the students. The rest of the module (16.5 hours) is devoted to practical work sessions including hands-on experiments on low-cost drones.

The next sections propose a focus on the content of this third module.

4 Step-by-step learning in simulation

Before experimenting with real drones, the first hour of practical work sessions is devoted to tutorials on ROS to learn its basics: developing a node in Python, publishing messages on topics, and using tools. It is done with a virtual machine provided to each student. After this first tutorial, students are invited to use the TUM ARDrone simulator[3] with Gazebo to pursue learning ROS basics and the development of robotic codes through a motivating drone application scenario. A pedagogical step-by-step approach has been developed to propose simple exercises to students in an incremental way, starting from very basic tests and finishing with the development of codes to fulfill a complete drone mission. This pedagogical approach is summarized by the flowchart presented in Figure 1.

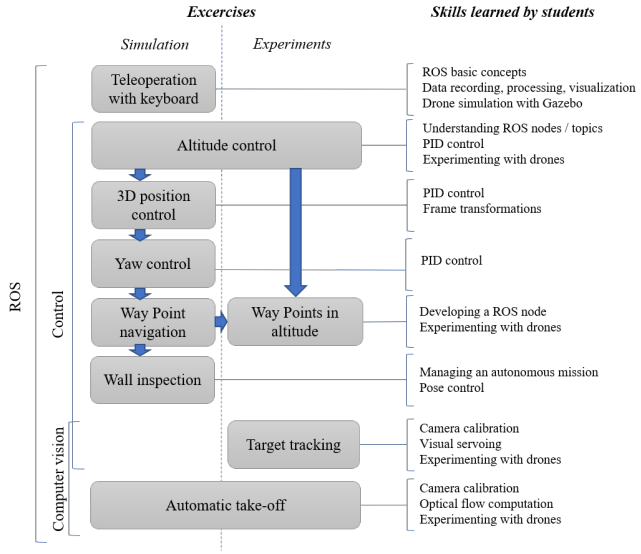

Fig. 1. Flowchart of the step-by-step pedagogical approach

[3] http://wiki.ros.org/tum_simulator

The first step is to familiarize students with ROS concepts, data visualization, message publication, etc. They are first invited to launch a teleoperation code and control the drone's motion with the keyboard. This is usually a motivating and funny part well appreciated by students. Nevertheless, they are asked to visualize and record sensor and localization messages. Recorded data can be replayed using the *rosbag* tool, and also post-processed by the students to produce time plots and trajectory plots. This first step aims simultaneously at safely familiarizing students with the drone they will use for experiments, and learning how to deal with the experimental dataset (recording, processing, visualizing).

In the next step, students focus on the development of a controller for the vertical motion of the drone. A ROS node developed in Python is provided to the students that have to complete the controller equations. Students usually develop a Proportional Controller to stabilize the drone at a given reference altitude, based on range measurements provided by a downward-facing range sensor. This code is tested and validated in simulation. It enables students to have a first glimpse at the structure of a ROS node, and understand how to manage input and output data through topics and messages. Once validated in simulation, this altitude controller is then tested by the students on a real drone (see next section).

Pursuing the step-by-step approach, the problem of 3D position control of the drone is then investigated by the students. The first exercise considers a simplified problem with zero yaw assumption. Therefore no transformation between reference frames is required for the computation of the control inputs (velocity commands). Once equations are implemented for the control of the (x, y, z)-coordinates of the position, a second exercise is proposed to generalize the developed code to handle non-zero yaw for the drone[4]. Simple coordinates transformation is then performed by the students to express the control inputs, initially computed in the inertial reference frame used for localization, to the body frame attached to the drone and which orientation depends on the current yaw of the vehicle. This exercise is completed with the development of a yaw controller. This part is validated in simulation, providing a reference pose to be reached by the drone.

The last step before working on the final wall inspection mission is Way Point Navigation. Students are invited to develop a ROS node in Python that will act as a Way Point Manager. Its role consists in sending reference position and yaw to the controller, depending on the actual position and orientation of the drone, and a list of predefined poses (3D position + yaw) to be reached. This step makes students develop a full ROS node on their own, taking as a starting point the Python code of the controller node, to be adapted and modified. A simple distance criterion (including on yaw error) is implemented by the students to trigger a switch to the next reference pose. The code is validated in simulation by the students on a simple trajectory (eg. a square pattern with arbitrary

[4] Pitch and roll motions are neglected in the design of the controllers which aim at computing velocity commands (translational velocity and yaw rate) to be applied to the drone, taking profit of the existence on-board of inner-loop controllers.

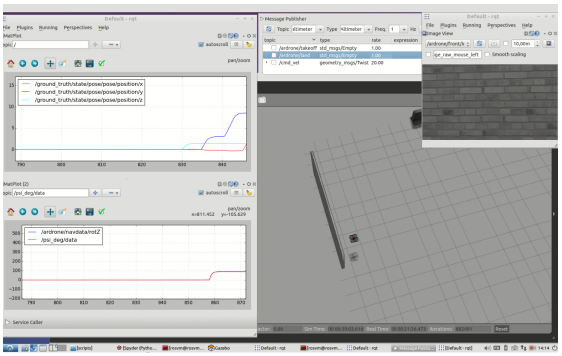

Fig. 2. Wall inspection mission in ROS Gazebo simulation. *Left*: time visualization of position and yaw, *Center*: Gazebo simulator, *Top-Right*: video from the simulated drone camera.

yaw references), and in experiments considering vertical motion only (see next section).

At this stage of the practical work sessions, the students have developed a set of ROS nodes implementing in Python a simple drone controller and Way Point manager that can be used for autonomous flight. The final exercise of this part is to address a wall-inspection mission autonomously by the drone. Students dispose of a simulation environment in Gazebo which includes a wall structure (see Figure 2). The objective is to make the drone take off, perform a visual inspection[5] of the two sides of the wall, and then return and land at the initial position. The mission must be realized in a fully autonomous way. Students are let free to decide how to parameterize this mission. Some of them use the teleoperated mode to control the drone with the keyboard to find adequate pose coordinates to be given to the Way Point manager. Other students proceed iteratively with tests and trials to get good coordinates directly using the simulation with the autonomous drone. Some groups of students defined the inspection trajectory with a rectangular shape around the wall at some constant altitude. Distance to the wall is sometimes adapted by some students to get the wall height inside the camera field of view. Other groups of students try to define some snake-like inspection patterns at different altitudes to improve coverage of each side of the wall. An example of realization can be seen in the video available at *https://tinyurl.com/2p9ed3fj*.

This step-by-step approach in simulation is completed by some experiments with real drones, as detailed in the next section.

[5] The video camera of the drone is required to provide images of the wall. But no specific requirement on coverage is specified in the exercise.

5 Hands-on experiments with drones

5.1 Drone Platforms

Making students perform experiments with real drones in classrooms implies different constraints. Safe platforms have to be used in a constrained and cluttered environment with people nearby (a classroom with students). The drone and associated equipment (batteries, spare propellers, computer, etc.) should be low-cost, as crashes may occur during the experiments. Software compatibility should be ensured between the drivers of the drone and the software environment as well as the programming language used for the labs (Ubuntu+ROS+Python).

In previous years, the ARDrone has been chosen. The existence of the TUM ARDrone simulator and ARDrone Autonomy library [8] for driver and control motivated this choice, in addition to the fulfillment of the three aforementioned constraints. Thanks to these characteristics, this low-cost platform has been indeed widely used both for academic research and teaching. Nevertheless, it is now discontinued, being no longer sold nor maintained.

Therefore, a transition to another platform has been decided this year and the Robomaster Tello Tallent (RMTT) drone has been selected. It is an update of the Tello Edu drone which provides new features such as an extension module for some additional sensors or RGB led matrix + front-facing range sensor. For the Tello Edu, there exist open-source ROS drivers developed by the robotics community. Nevertheless, at the time of this year's lab sessions, no drivers were found for the RMTT, i.e. that include the extension module for ROS1. Therefore a specific driver has been developed, that provides a wrapping of the Python Robomaster SDK[6] from DJI and enables access via ROS topics to sensors, battery, control inputs, take-off and landing, mission pads (colored tags provided with the drone), and extension module (RGB LED, LED array, front-facing range sensor).

This driver is freely available[7] and work is currently being done to finalize the integration of a localization system based on the mission pads provided with the drone. The first step already done consists in using a *carpet* of mission pads with known relative locations to compute the 3D position of the drone during flight. An example is given in Figure 3, showing the mission pads on the left and the localization of the pads and drone during the flight on the right. Taking one of the pads as a reference (pad no 1, at the center, in Figure 3), the real-time position of the drone is computed in that reference frame. Using multiple pads enables to increase the volume where the drone can be localized.

Such a localization system will be useful to make students develop and experiment with control algorithms in the classrooms, without the need for an expensive motion-capture system or a specific setup with other types of markers [9] than the ones already provided with the drone. It will also be convenient for new exercises that will be proposed to the students in the close future, such as information fusion for localization (eg. Kalman filtering using accelerometers

[6] https://github.com/dji-sdk/RoboMaster-SDK
[7] https://github.com/bertrandsylv/rmtt_ros_driver

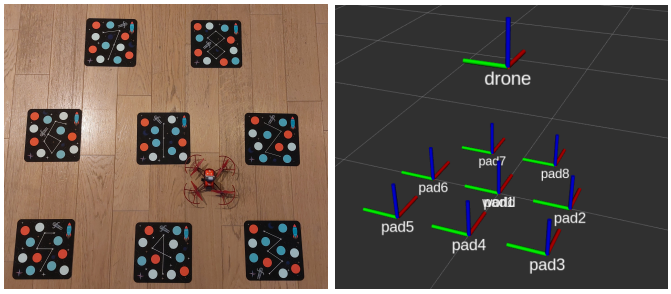

Fig. 3. *Left*: mission pads and drone before take-off, *Right*: Visualization in RViz of positions of pads and drone during flight

and position measurements from the pads) or Simultaneous Localization and Mapping (SLAM).

Classroom experiments already realized by students in the context of this curriculum are detailed in the next section.

5.2 Experiments

Several hands-on experiments are proposed to students as exercises as shown in the flowchart of the pedagogical approach of this curriculum (Figure 1). These experiments are done in the classroom during the lab sessions. Students always enjoy testing their algorithms on real flying drones. It is very motivating for them, and this is the reason why experiments have been introduced at different steps of the practical work sessions. Students can record videos of their experiments and get experimental data as well.

The first and most simple experiment consists in controlling the vertical motion of the drone. Controllers developed by the students are tested and validated, each group using its own controller tuning. Figure 4 shows vertical trajectories (altitude recorded from the bottom range sensor) realized by different groups of students with the RMTT drone. The two plots at the center show a good closed-loop behavior of the drone, whereas the ones on the left and the right correspond to oscillating behaviors obtained for badly tuned controllers. For the plot on the right of Figure 4, the drone oscillated before diverging to the ceiling of the classroom. Observing the drone behaviors during the flights made the students more easily understand the concepts behind automatic control (stability, damping, steady state error, etc.). After each group has performed its experiment, a debriefing with all the students has been done to compare and discuss together results and ways of improvement.

A direct extension of the previous experiment proposed to the students is altitude Way Point navigation, for which the drone has to reach and stabilize successively at points of different altitudes.

A third experiment proposed to the students concerns target tracking using image-based information. It has been realized in the past years with the

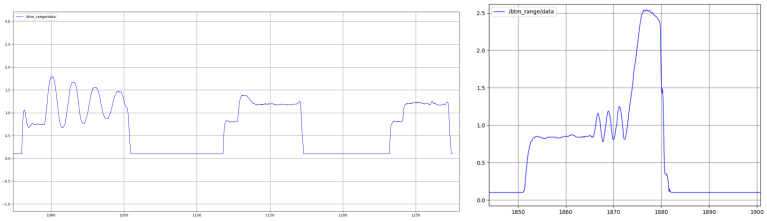

Fig. 4. Closed-loop vertical motion of RMTT drone for four different gain tunings of PI controller (flight experiments by the students).

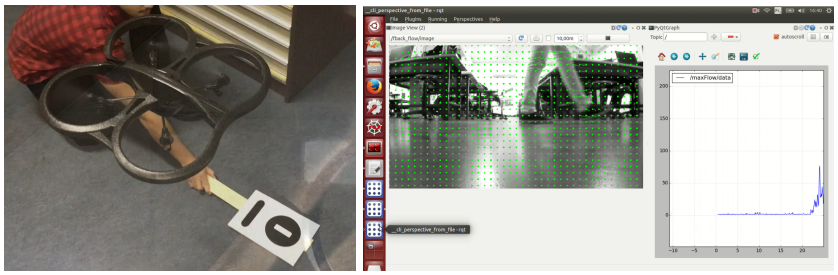

Fig. 5. *Left*: Visual servoing for tracking of a ground mobile target, *Right*: Optical flow computation for automatic take-off triggered by motion detection

ARDrone, taking profit from its tag recognition capability. A control law (visual servoing) is developed by the students to move the drone such that the detection of the target remains centered in the image provided by the bottom camera (see left part of Figure 5). The same experiment will be done with the RMTT drone in the new version of the exercises.

Finally, to draw a link to the *Perception and computer vision for mobile robotics* module of the curriculum, the last part of the practical work sessions is devoted to an exercise involving computer vision. The objective is to develop a ROS node enabling automatic take-off of the drone when motion is detected in the image of its front camera. This could correspond to a surveillance scenario where a drone would take off if an intruder is detected. Optical flow monitoring is given as proposal of a solution for motion detection. A rosbag with a video record of someone walking in front of the landed drone is given to students for developing and testing their solution. Once tested, each group can validate by experimenting on the real drone. An example is given on the right part of Figure 5 showing the video frame with optical flow visualization (left) along with a metric proposed by the students (maximum norm of the optical flow) for motion detection (right). The experiment can also be seen in the video available at *https://tinyurl.com/2p9ed3fj* . For time reasons, optical flow computation is realized using the ROS *opencv_apps* package[8]. A demonstration of camera calibration is also done in the classroom with all the students, before experiments.

[8] http://wiki.ros.org/opencv_apps

6 Student projects with drones

In parallel to this curriculum, some groups of students also work on projects, such as the one in the context of the Cap Project initiative (see [7]). Some projects are dedicated to drones. Some examples over the past years are dot painting using single or multiple drones, designing a low-cost drone, participating in student drone contests, etc. These projects are managed by teachers and advisers in a complementary way to the curriculum to ensure that students can benefit from both.

An example of project is face and emotion recognition with drones. It aims to make the drone recognize the emotion of the person in front of it and interact accordingly through movements. The drone used in this project is the RMTT, with which we communicate using a Python interface that includes Tello libraries provided by the Tello SDK. Video frames captured by the drone are sent to the computer, which recognizes the emotion of the person in front of the drone and sends movement commands, so the drone reacts to the emotion. Social interaction in this project is based on four main tasks: face detection, face tracking, emotion recognition, and drone reaction.

Face detection is done using the OpenCV library. The computer receives the video stream sent by the drone and applies the OpenCV Haar Cascade classifier to detect faces in a video frame. Various face detection models are available in the literature such as OpenCV, SSD, MTCNN, and RetinaFace. In the context of this project, face detection has to be as fast as possible with acceptable accuracy. MTCNN and RetinaFace are very powerful face detectors in terms of accuracy. However, they require a long execution time, which makes them not suitable for real-time face detection. On the other hand, OpenCV and SSD face detectors proved to be faster at the cost of a slight accuracy degradation. This is why the OpenCV face detector was used in this project.

Drone movements are impacted by two main mechanisms: face tracking and reaction to the captured emotion. Face tracking consists in maintaining the detected face centered in the captured video frames. To do so, after capturing a video frame by the computer and detecting the face position in it, the difference between this position and the center of the frame is calculated and movement commands are sent to the drone so it slides horizontally (right/left) and/or vertically (up/down) so the next captured face position is the closest possible to the video frame center. This is done using PID (proportional-integral-derivative) speed controllers.

Emotion recognition is done using the open-source DeepFace Python library. This library offers several features such as face verification (comparing two faces), face recognition (finding a known face in a frame), and facial attribute analysis (age, gender classification, and emotion analysis). In this project, only the emotion analysis feature is used. Emotion recognition in DeepFace is based on a combination of convolutional and dense neural network layers. In [10], authors tested twelve emotion recognition models, all based on Deep Learning and Convolutional Neural Networks (CNN) using the CK+ (extended Cohn-Kanade) [11] and the Fer2013 (Facial Expression Recognition 2013) [12] datasets. They showed

that the DeepFace algorithm was the most accurate in emotion recognition. Another interesting feature of DeepFace is that it is a lightweight system, which makes it suitable in the context of this project since emotion recognition time does not have a significant impact on the smoothness of the video transfers between the drone and the computer. The DeepFace system allowed for the recognition of six main emotions (anger, disgust, fear, happiness, sadness, and surprise) and neutrality.

As a reaction to the emotion of the person in front of it, the drone uses two mechanisms. First, it displays a smiley on its RGB-led matrix that mimics the recognized emotion. Figure 6 shows how emotions are displayed on the drone's RGB-LED matrix. Second, the drone makes some motions as a reaction to the emotion of the person in front of it. This reaction can be of two types: 1) showing empathy by making motions that interpret the detected emotion, and 2) making motions that try to change a negative emotion of the person.

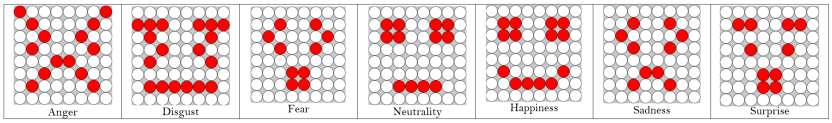

Fig. 6. Emotions displayed on the drone RGB-LED matrix

Some of the students involved in this project also enrolled in the robotic curriculum presented in this paper. Using the same drone for the project as in the curriculum helps to make it profitable for students. As the project started before and finishes after the curriculum, students may also find a special interest in lectures, tutorials, and lab sessions, directly implied by practical concerns/issues raised from their project.

7 Conclusions

In this paper, some feedback has been presented on a short curriculum on robotics developed for last year's students at ESIEA, a graduate school of engineering. One of the characteristics of this curriculum is to propose hands-on experiments in classrooms during practical work sessions with low-cost drones.

The content of the curriculum has been presented, with a specific focus on one of its modules devoted to ROS and computer programming for robotics. Examples of experiments realized by the students as exercises integrated into the work sessions have been provided. This practical aspect is usually well appreciated by the students and helps to maintain their motivation and attention. Seeing, understanding, and analyzing practical results help them to have deeper looks at theory.

The next steps will concern the experimental setups, to be able to provide one drone per group of students with a localization system based on mission pads, as well as the development of new experiments and exercises regarding sensor fusion for localization and SLAM.

References

1. Yeppes, I. and Barone, D.A.C. and Porciuncula, C.M.D, "Use of Drones as Pedagogical Technology in STEM Disciplines", Informatics in Education, vol. 21, no. 1, pp. 201-233, 2022.
2. Beuchat, P.N. and Sturz, Y.R. and Lygeros, J., "A Teaching System for Hands-on Quadcopter Control", IFAC-PapersOnLine, vol. 52, no.9, pp.36-41, 2019.
3. Canas, J.M. and Martin-Martin, D. and Arias, P. and Vega, J. and Roldan-Álvarez, D. and Garcia-Pérez, L. and Fernandez-Conde, J., "Open-Source Drone Programming Course for Distance Engineering Education", Electronics, vol. 9, no. 2163, 2020.
4. Bertrand, S. and Marzat, J. and Stoica Maniu, C. and Makarov, M. and Filliat, D. and Manzanera, A., "DroMOOC: a Massive Open Online Course on Drones and Aerial Multi Robot Systems", 12th UKACC International Conference on Control, Sheffield, UK, 2018.
5. Brand, I. and Roy2, J. and Ray, A. and Oberlin, J. and Tellex, S., "PiDrone: An Autonomous Educational Drone using Raspberry Pi and Python", IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 2018.
6. Invernizzi, D. and Panza, S. and Giurato, M. and Yang, G. and Chen, K. and Lovera, M. and Parisini, T., "Integration of Experimental Activities into Remote Teaching using a Quadrotor Test-bed", IFAC Workshop on Aerospace Control Education, Milano, Italy, 2021.
7. Bertrand, S. and Prevost, L. and Ionascu, F. and Briere, A. and Koskas, R. and Taquet, R. and Andrianantoandro, F.and Kanzari, M., "Light Painting with Mobile Robots as Motivating Projects for Robotics and Control Education", 13th International Conference on Robotics in Education, Bratislava, Slovakia, 2022.
8. Monajjemi, M. , AR Drone Autonomy library, http: //wiki.ros.org/tum_ardrone, 2014.
9. Kayhani, N. and Heins, A. and Zhao, W. and Nahangi, M and McCabe, B. and Schoellig, A. P., "Improved Tag-based indoor Localization of UAVs using Extended Kalman Filter", 36th International Symposium on Automation and Robotics in Construction, 2019.
10. Chiurco, A., Frangella, J., Longo, F., Nicoletti, L., Padovano, A., Solina, V., Mirabelli, G. and Citraro, C., "Real-time Detection of Worker's Emotions for Advanced Human-Robot Interaction during Collaborative Tasks in Smart Factories", Procedia Computer Science, pp. 1875-1884, 2022.
11. Lucey, P., Cohn, J., Kanade, T., Saragih, J., Ambadar, Z. and Matthews, I., "The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression", 2010 IEEE Computer Society Conference On Computer Vision And Pattern Recognition-workshops, pp. 94-101, 2010.
12. Goodfellow, I., Erhan, D., Carrier, P., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., Lee, D. et al., "Challenges in representation learning: A report on three machine learning contests", International Conference On Neural Information Processing, pp. 117-124, 2013.