Integrating Experimental Data Sets and Simulation Codes for Students into a MOOC on Aerial Robotics [⋆]

S. Bertrand [*] J. Marzat [*] G. Le Besnerais [*] A. Manzanera [**]
C. Stoica Maniu [***] M. Makarov [***]

[*] *ONERA-The French Aerospace Lab, Palaiseau, France (e-mail: {sylvain.bertrand; julien.marzat;guy.le_besnerais}@onera.fr).*
[**] *ENSTA-ParisTech/U2IS, Palaiseau, France (e-mail: antoine.manzanera@ensta-paristech.fr)*
[***] *Laboratoire des Signaux et Systèmes (L2S), CentraleSupélec-CNRS-Université Paris-Sud, Université Paris-Saclay, 3, rue Joliot Curie, 91192 Gif-sur-Yvette, France (e-mail: {cristina.stoica;maria.makarov}@l2s.centralesupelec.fr)*

Abstract: This paper addresses the integration of experimental data sets and simulation codes into a MOOC. Such additional material is both used in the lecture videos and proposed to the students as a complement to put into practice the theoretical notions presented. The chosen pedagogic approach hence relies on a tight integration between these elements and is illustrated through the example of DroMOOC, a MOOC on aerial robotics.

Keywords: e-learning, MOOC, open educational resources.

1. INTRODUCTION

Massive Open Online Courses (MOOCs) are getting more and more popular and followed by students either to discover new topics not addressed in their curricula or as a complement to classical courses they are following. MOOCs are also sometimes imposed by teachers as a pre-requisite to students, especially for project based-learning or inverted classrooms.
Although more in phase with digital habits of modern students, online courses are mainly based on videos which makes of them, in their pedagogy, the direct continuity of one-directional traditional lectures. Some MOOCs try to provide complementary materials such as exercises, simulation codes, etc.
As practice is recognized as a good way to transform acquired knowledge into new skills, it is of a huge importance to be able to complement MOOCs by active practical work for the students. In the field of robotics, this consideration makes even more sense since hardware practice, computer programming, experimental testing are skills that are required for roboticists and hence expected from young professionals. A main issue is therefore how to combine remote teaching and practice for students who follow a MOOC. This aspect is addressed in this paper through the example of DroMOOC [1], a MOOC on aerial robotics.
DroMOOC is a MOOC dedicated to multirotor drones and aerial multi robot systems (Bertrand2018). It is realized by research scientists, engineers and professors from three French research and education institutions: ONERA - the French AerospaceLab, CentraleSupélec (Laboratory of Signals and Systems) and ENSTA - two graduate engineering schools. This MOOC is intended to students and young researchers. Its main objective is to provide them theoretical knowledge on this field but also skills that would fit the needs and recruiting requirements of industries, startups, research laboratories working on drones.
Topics addressed by DroMOOC cover dynamic modeling, sensor fusion and state estimation, computer vision, environment modeling, motion planing and control. Courses are proposed at basic and advanced levels. The basic level presents simple notions that are accessible to students with an elementary scientific background and that can easily be applied in practice to a real drone (e.g. for a student project). This basic level is a full standalone course. The advanced level addresses more complex methods that are closer to the state of the art of research or R&D in industry. It aims at giving to the students skills that can be professionally valorized (e.g. for a PhD, a job, etc.) and applied in practice to a real robot leading to improved performances compared to the basic level.
In complement to videos experimental data sets and simulation codes are provided to the students to put into practice the notions of the courses. Experimental data sets are realized using professional hardware and facilities of the COPERNIC Lab at ONERA [2]. Simulation codes are also realized aiming at being close to professional standards used in robotics.

[⋆] The DroMOOC e-learning project has been funded by *AAP IDEX 2017 Paris-Saclay, Numérique pour la formation à distance*, in the context of *Université Paris-Saclay & Programme d'Investissements d'Avenir* from French State and ANR.
[1] www.onera.fr/dromooc

[2] www.onera.fr/copernic

Remotely providing such material to students of a MOOC arises several issues and difficulties for the teachers which can be listed as follows:

- Find a way to integrate tightly this material to the video content of the MOOC (i.e. both within the videos and as complement to lectures).
- Find a good trade-off between professional level of the data sets and codes (robust but complex) and pedagogic approach for the students (readability).
- Provide materials that can be used as if they were directly related to a robotic hardware platform.
- Provide materials that can be used by students without requiring a too specific computer equipment nor a strong computer science expertise.

This paper provides some insights related to these issues, through the example of DroMOOC. The next section is devoted to the ways this additional material can be provided to the students. Section 3 focuses on experimental data sets whereas Section 4 addresses the case of simulation codes. Concluding remarks are given in the last section.

Note that focus is made on estimation and control in this paper although DroMOOC aims at developing the same approach for the other topics it addresses.

2. ENVIRONMENT MADE AVAILABLE TO THE STUDENTS

2.1 ROS environment for remote teaching

This complementary material (data sets and simulation codes) is integrated into the ROS (Robotic Operating System) environment. ROS is well known in robotics and widely used both by academics and industry. This choice hence enables the students to benefit from the practice of this software which is widely professionally recognized.
ROS is a powerful tool but can be complex to install and to be used by students who do not have already an experience with it or with a limited background in computer sciences. Therefore an important objective of DroMOOC is to make the use of ROS as simple as possible. The main difficulty is that it relies on specific code structures, data types and messages, etc. that would require an entire dedicated course for the students, as it is the case in most of the robotics curricula.
The paradigm chosen in DroMOOC is to limit as much as possible the required expertise in ROS for the students by providing them data, scripts and codes easy to be used and to be completed and by using Graphical User Interfaces (GUI) as much as possible. Running codes in a Linux environment, as required for the execution of ROS, usually requires complex command lines to be entered in terminals, which can be prohibitive for some students. The use of a dedicated GUI integrated in ROS[3] is therefore proposed instead. Several GUIs are also used for visualization and parameter tuning, based on specific layouts designed by the teachers of DroMOOC, to facilitate the experience with ROS.
In fact the objective is twofold: make the provided material usable by students who are not expert in ROS, and help the students to discover ROS and use it for the design of robotic control systems.
The choice of the programming language used in DroMOOC follows the same paradigm. Python has been chosen (instead of C++ which is also supported by ROS) for a greater simplicity of use by the students who may not be experts in computer programming. By preparing Python scripts, the teachers of DroMOOC help the student to focus on the direct application of the scientific content of the MOOC, minimizing the issues related to computer programming which can be time consuming, especially without teachers being physically present.

In robotics it is important to implement and make experiments with real hardware, which is not an easy task for a MOOC as one can not provide robotic platforms to the students as in classical labs. In DroMOOC, the use of ROS also aims at fulfilling this objective. Indeed, as a middleware for robotics, ROS enables connection to real hardware and data recording. It is therefore possible for the teachers to record data sets from sensors (eg. IMU of a drone) and make them available to the students for time replay as if data were read directly from the real sensors. Another huge advantage is also that the code provided by the teachers and/or developed by the students for processing these data can both be run in simulation/replay and in connection to real hardware. Student can then use the codes developed in the context of DroMOOC for practical projects in robotics.

2.2 Distribution of the environment to the students

Another difficulty for MOOCs, when compared to classical labs where computers can fully be prepared by the teachers, is to provide the students with digital resources without requiring from them a too specific computer configuration. Especially for ROS, a Linux distribution is required and its installation can be prohibitive to students that are not used with it and/or do not wish to modify their computer configuration just to follow a MOOC.
The envisaged solution will consist in making available the code and data sets through different ways that will enable as many students as possible to participate:

- For students that are able to install Linux and ROS by themselves, data sets and codes are made available on a Github repository along with installation instructions.
- For students that have a Windows computer and are not used to Linux and do not want to modify their personal installation on their computer, a Virtual Machine runing Linux and ROS is proposed for download.
- For students who do not have a computer or are interested in a light alternative platform, a disk image for Raspberry Pi is also proposed for download. This implies constraints on the computational power available for running the codes.

Some tutorial videos will be provided to the students to explain the basis on the use of the environment and of ROS. The main idea is to avoid doing a long course on ROS and focusing only on information strictly required for the use of the data sets and simulation codes that are provided. These videos are part of the online content of the MOOC but will also be present in the distributed environment to

[3] wiki.ros.org/node_manager_fkie

allow for offline uses. In addition to these tutorial videos PDFs files will be provided for each exercise or project both on the website of the MOOC and in the distributed environment.

3. EXPERIMENTAL DATA SETS

3.1 Nature of provided data sets

The experimental data sets to be considered in DroMOOC are

- measurements from Inertial Measurement Units (IMU) recorded during flight and with ground truth provided by a motion capture system (for multi-sensor fusion and state estimation course);
- videos recorded from on-board cameras of drones during the flight (for computer vision course);
- 3D maps of environments build by the drone during the flight (for SLAM and environment modeling courses);
- position measurements of several drones flying in formation (for multi-vehicle localization course).

These data are recorded using professional equipment and facilities and can be easily exploited by the students (ROS format for data replay and exploitation scripts as mentioned in the previous section). An example of data set for attitude estimation is presented in Figure 1.

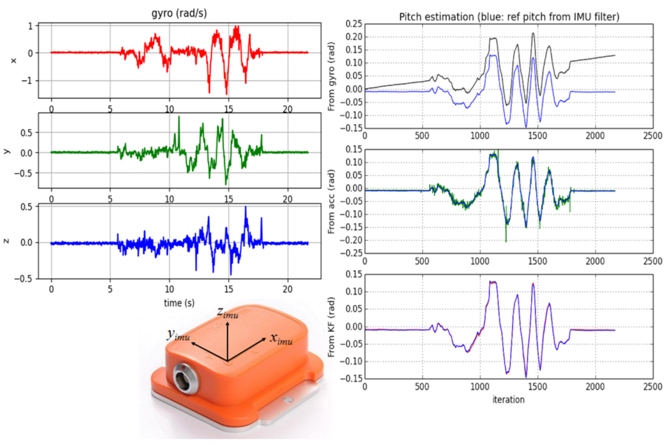


Fig. 1. Example of IMU data provided to the students and associated estimation results

Enabling students to get hands on professional data is valuable for them in terms of experience. The main difficulty is that professional data can be complex to handle and the link to notions explained in the lectures can be difficult to be seen. Therefore it has been chosen, when possible, to keep only parts of the data that are strictly required by the students for understanding and practice, and rename the data in an understandable way with direct link to the courses. More specifically, in ROS, data are encapsulated and exchanged through messages for replay and processing. Messages are based on data types which are very informative but can be complex and sometimes not intuitive for students that are not used to ROS. Therefore it has also been chosen to build on the capability offered by ROS to define user's message data types. For example in a course of attitude estimation, it is more intuitive for students to develop a state estimator by using first an attitude representation based on Euler angles than on quaternions, which is a more complex notion for the students. Since attitude representation is mainly based on quaternion messages in ROS, specific message types have be defined for DroMOOC to represent Euler angles and hence facilitate the access to the data for the students.

3.2 Integration in the pedagogic approach

This additional material for practice (data sets) has to be integrated in the curriculum of the MOOC. A first possibility would consist in designing exercises and instructions for practical sessions based on this material and introduced to the students by specific videos or documents. But to stimulate the interest of the students, it has been chosen, when possible, to integrate in a more tightly way this material to the videos of the lectures, which are the first elements to be seen by the students.
The example of multi sensor fusion and IMU measurement data set is taken to illustrate this point. A flow chart of the pedagogic approach is illustrated on Figure 2. A first video introduces the interest of fusing measurements from multiple sensors. More precisely the problem of pitch estimation of a drone from accelerometer and rate gyro measurements is introduced by presenting the results based on processing the experimental data.
A second video introduces Kalman filtering as a fusion method and provides theoretical details to the students so that the methodology can be applied to get the results presented in the first video. At this step, a code (Python script) is given to the students with the full implementation of the Kalman Filter based approach so that they are able to reproduce directly the results obtained and presented in these first two videos. It enables the students to examine the implementation of the approach, and understand the tuning of the parameters by trying and testing with the code on the experimental data.
Then, a third video is proposed to the students to address the estimation of the full attitude (roll, pitch and yaw) of the drone. The results presented to the students are derived from the same experimental data set. The code is not given to the students but, after this video, they are invited to develop their own codes to produce the same result. This can be built directly upon the code that has been made previously available to them, and they dispose in the last video of the result that should be obtained. In this way, the students are given the opportunity to put into practice by developing their own code in complete autonomy, but with controlled starting and ending points in the approach (code basis and results to be obtained).
It is worth mentioning that the whole approach is based on the use of the ROS data replay system (rosbag) which enables a replay of the data in conditions close to 'real time', i.e. as if the measurements to be processed were the ones delivered in reality by the IMU. Therefore the code given to the students and to be completed by them operates in conditions similar to the one of an architecture embedded on board of a drone. In addition, taking advantage of the standardization of the types of data and messages used by ROS, the students can use their code for practical implementation on a real robot (running ROS), e.g. for

a student project, resulting in a concrete production by the student.

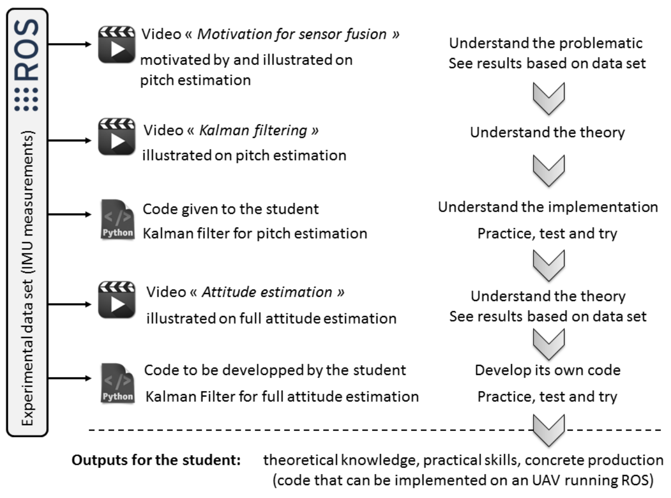

ROS
Experimental data set (IMU measurements)
Video « Motivation for sensor fusion » motivated by and illustrated on pitch estimation
Understand the problematic
See results based on data set
Video « Kalman filtering » illustrated on pitch estimation
Understand the theory
Python
Code given to the student
Kalman filter for pitch estimation
Understand the implementation
Practice, test and try
Video « Attitude estimation » illustrated on full attitude estimation
Understand the theory
See results based on data set
Python
Code to be developped by the student
Kalman Filter for full attitude estimation
Develop its own code
Practice, test and try
Outputs for the student:
theoretical knowledge, practical skills, concrete production
(code that can be implemented on an UAV running ROS)

Fig. 2. Flow chart of an example of tight integration of experimental data and codes with the video content of the MOOC.

4. SIMULATION CODES

4.1 Drone simulator

Practice of control algorithms is much more difficult as it would require implementation and tests on a real hardware platform. Note that open and remote experimental robotic testbeds exist, such as the Robotarium project by GeorgiaTech (Pickem2017) which could be used as a complement to a MOOC. In the context of aerial robotics, it is not possible to develop this type of resource. Therefore, for MOOCs and remote teaching, simulation remains a way to put into practice control engineering notions. Using ROS enables the students to develop codes in simulation that could also be used on real platforms, as more and more robots (education, research, industry) are now operated using this middleware. Therefore a drone simulator has been considered for DroMOOC.
There exist such simulators in the robotic community, eg. (Furrer2016). Nevertheless, for students that do not already have a good background with ROS, using these codes could be too complex or it could be time consuming for the students to use it as a simple complement of a MOOC. In addition, the direct parallel between control engineering notions presented in the lecture videos and the code may not be easily seen. Another concern is that students should be able to quickly see and understand the structure of a simulation and the implementation of a dynamic model without being an expert in computer programming. For control system curricula it is indeed important for students to learn estimation and control methods but also the basis of the simulation of a dynamic system.
Therefore a very simple and light simulator has been developed for DroMOOC that aims at facilitating the direct correspondence between theoretical notions in control engineering and practical implementation. It is composed of a simple ROS node (i.e. code executed at a given sampling period, node denoted *quadrotorSimu* in Figure 3) written in Python which integrates numerically the equations of motion of a multi-rotor drone, given its control input values, and outputs the state of the system. Using the RVIZ visualization tool integrated to ROS, a 3D visualization of the drone is also proposed to the students (Figure 4) which is a didactic and intuitive way of visualization for them, in addition to classical time plots. Contrary to other existing simulators running more sophisticated simulation engines (such as Gazebo), the simulator developed for DroMOOC is kept voluntarily simple and requires less computational power, which enables compliance to a use on a low cost Raspberry Pi computer for example (see Section 2.2).
A simulator for multi-drones is also considered for DroMOOC on the same basis and with the same pedagogic objectives.

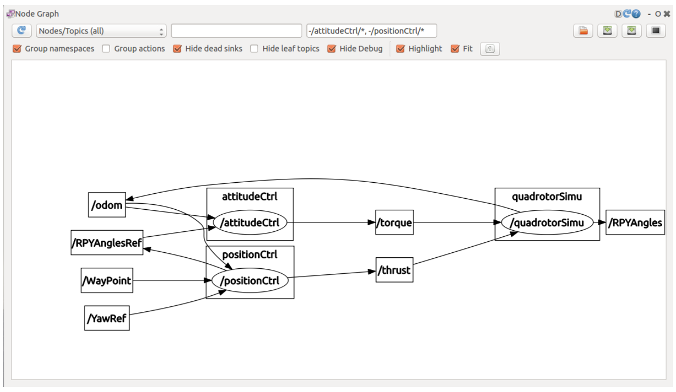
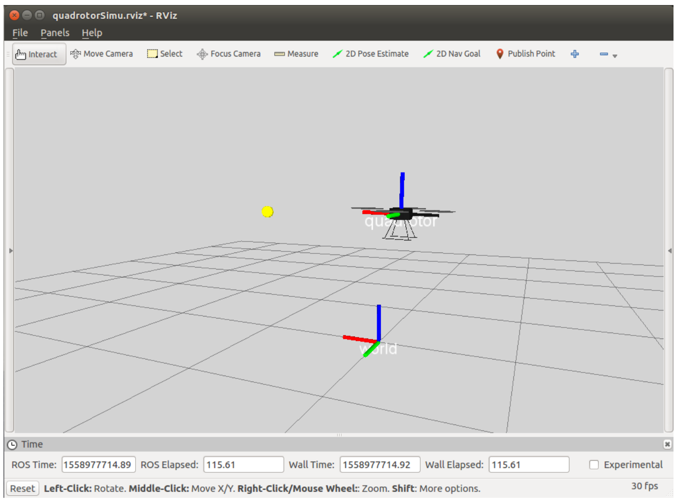


Fig. 3. Structure of the developed ROS simulator, with simulated dynamic model (*quadrotorSimu* node) and controllers (*attitudeCrtl* and *positionCrtl* nodes). (Graph automatically generated from code using ROS NodeGraph)



ROS Time: 1558977714.89 ROS Elapsed: 115.61 Wall Time: 1558977714.92 Wall Elapsed: 115.61

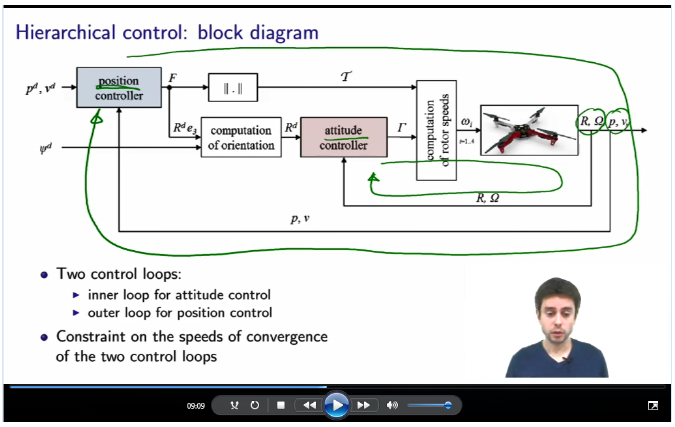
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

Fig. 4. 3D visualization of drone flight simulation and Way Point (in yellow) for position stabilization. (ROS RVIZ)

4.2 Control algorithms for drones

Teaching control of aerial robots requires to introduce to the students different notions of control theory such as state space representation, stability of dynamic systems, classical control algorithms (PID, state feedback, LQR,

etc.). Some elements should also be presented to the students regarding the cascaded structure of controllers which are usually implemented on drones (attitude control - position control).
In DroMOOC, this last point is presented to the students in an introduction video (see video screenshot in Figure 5). As classically in a lecture, mathematical notations and variable names are introduced and block diagrams are used to illustrate graphically the structure of the controller and the closed loop system.
In most of existing simulators or simulation tools, excepted from Matlab Simulink, such a graphical representation is hard to be found by the students and the link between a simulation code and these notions learned from the lecture may not be directly intuitive. In the simulator developed for DroMOOC, one takes advantage of the code structures offered by ROS and of its visualization tools. Indeed, different codes (or *nodes* in ROS) can be developed for attitude and position controllers and linked to the simulation code of the drone dynamic model. Information exchanged between these nodes (or *topics* in ROS) can be renamed to follow the notations introduced in the lecture. Finally, the students can also obtain the graphical structure of Figure 3 of their code, automatically generated by ROS (ROS NodeGraph). A parallel to the lecture is hence facilitated for them.

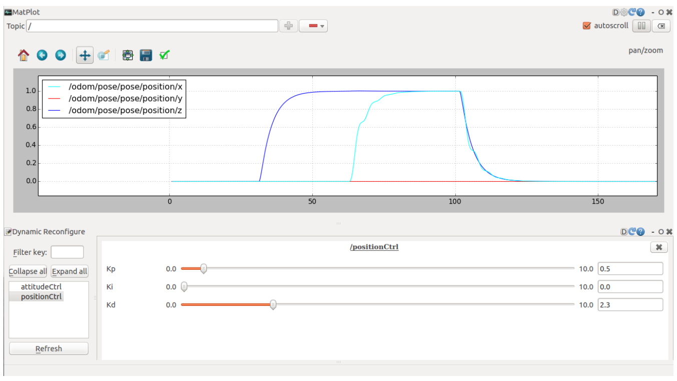

Hierarchical control: block diagram

- Two control loops:
  - inner loop for attitude control
  - outer loop for position control
- Constraint on the speeds of convergence of the two control loops

Fig. 5. Controller structure as presented to the students in a lecture video of the MOOC.

4.3 Integration in the pedagogic approach

As for data sets, simulation codes have to be integrated in the curriculum of the MOOC and a tight integration is chosen in DroMOOC. The example of PID control is presented in this section to illustrate this point.
A fist video is proposed to the students as an introduction to the problem of controller design for a drone. Cascaded structure of usual controllers for drones is presented, in correspondence to the structure of the simulation code that is provided as additional material to the MOOC (see previous section).
A second video, part of the basic level, introduces the problem of attitude control and the design of a simple PID control law assuming low angles. A link to a video of the advanced level is made regarding extension to nonlinear attitude control. Students are also invited to have a look at the provided Python code implementing a PD controller for attitude stabilization.
Using the capacity of ROS to modify online (here, while the simulation is running) the value of some parameters (ROS dynamic parameters), the students can tune the coefficients of the control law. Performing practical gain tuning is indeed a good way for students to understand PID control in a complementary way to theoretical fundamentals. A Graphical User Interface based on ROS RQT plugin is proposed to do so (see Figure 6 for position stabilization) and to visualize the influence on the closed loop system. Time plots are available on the same interface so that the students can use it as a standalone tool. They can also have a look simultaneously at the 3D visualization of the flight simulation (Figure 4) to assess the behavior of the drone.


MatPlot
Topic /

Dynamic Reconfigure
Filter key:
attitudeCtrl
positionCtrl

/positionCtrl

Kp	0.0	10.0	0.5
Ki	0.0	10.0	0.0
Kd	0.0	10.0	2.3

autoscroll
pan/zoom
/odom/pose/pose/position/x
/odom/pose/pose/position/y
/odom/pose/pose/position/z
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
Collapse all
Expand all
Refresh

Fig. 6. Graphical User Interface proposed to the students for online PID tuning and simultaneous visualization of its effects on the closed loop system. (ROS RQT)

Only a PD controller is implemented in the code provided to the students. They are also invited to develop an integral action to understand more deeply its effect and the related issues from an implementation point of view.
They are also invited to do the same with a Python code implementing a PD control law for the position stabilization problem. Online gain tuning offers the opportunity to the students to try different values for the gains and understand their influence and the constraints related to a cascaded controller structure (convergence speed ratio required between inner loop for attitude control and outer loop for position control).
Finally, a third video of the basic level is proposed to the students for the position control problem, but presenting a LQR control approach. This controller is intentionally not provided to the students, but some guidelines are given to them, so that they can develop it by themselves, in an ”active” learning way. Performance comparison is made possible to them with respect to PID control, since this code has already been used by the students.

Other types of controllers (e.g. Model Predictive Control) are considered in the advanced level of the MOOC.

From this full basis provided to the students, a ”project” can also be proposed to them to develop and test with the simulator a way point navigation strategy for the drone and to extend to trajectory tracking all the methods seen previously for a stabilization/regulation problem.

Therefore the proposed pedagogic approach relies on a combination and tight integration of classical lectures proposed through videos and active participation of the students through code development in simulation. The pedagogic approach developed here results in a flow chart similar to the one of Figure 2.

Finally it is worth noticing that the code proposed to and developed by the students can be used by them as a basis for the control of a real drone running ROS, with possibly slight modifications required regarding the types of inputs/outputs related to hardware/software characteristics of the robot.

5. CONCLUSIONS

Convinced by the fact that practice is required by students in the process of learning, integration of experimental data set and simulation codes has been investigated in the realization of DroMOOC, a MOOC on aerial robotics and multi robot systems realized by ONERA, ENSTA and CentraleSupélec. A tight integration between this additional material and lecture videos has been performed when possible, aiming students to develop skills, practical experience and get concrete realizations that could be implemented on physical hardware systems.

A feedback will be asked to students regarding the MOOC (its content, structure, etc.) and more specifically on the relevance and interest of this integration of data sets and simulation codes to more traditional video content. This feedback will be used to assess the success of this "active" learning approach among the students and to improve both the proposed content and the pedagogy.

Validation of the MOOC will be first done by a multiple choice questions and answers quiz. The score obtained will be indicated on a certificate that will be sent to the student. Making available data sets and codes in the proposed integrated environment would enable to design remote assignments. As for benchmarks or contests in the robotics community, evaluation can be made by the way of specific performance metrics developed by the teachers and given to the students along with evaluation scripts implementing them. Students can hence auto-evaluate their work, while developing their project, before final submission to the teacher. The same scripts will be used by the teacher for attributing (part of) the mark. This will be considered in a future evolution of the MOOC.

REFERENCES

S. Bertrand, J. Marzat, C. Stoica Maniu, M. Makarov, D. Filliat and A. Manzanera, "DroMOOC: a Massive Open Online Course on Drones and Aerial Multi Robot Systems", *12th UKACC International Conference on Control* Sheffield, UK, 2018.

D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed", *IEEE International Conference on Robotics and Automation* Singapore, 2017.

F. Furrer, M. Burri, M. Achtelik and Roland Siegwart, "Robot Operating System (ROS), The Complete Reference (Volume 1)", Chapter 23, *Studies Comp. Intelligence Volume Number:625*, 2016.