Draft Version. Final version published in INTED2017

EXPERIENCES ON A MOTIVATIONAL LEARNING APPROACH FOR ROBOTICS IN UNDERGRADUATE COURSES

J.R. Ruiz-Sarmiento, C. Galindo, J. Gonzalez-Jimenez

Machine Perception and Intelligent Robotics Group, System Engineering and Auto. Dept., Instituto de Investigación Biomédica de Málaga (IBIMA), University of Málaga (SPAIN)
jotaraul@uma.es, cipriano@ctima.uma.es, javiergonzalez@uma.es

Abstract

This paper presents an educational experience carried out in robotics undergraduate courses from two different degrees: Computer Science and Industrial Engineering, having students with diverse capabilities and motivations. The experience compares two learning strategies for the practical lessons of such courses: one relies on code snippets in Matlab to cope with typical robotic problems like robot motion, localization, and mapping, while the second strategy opts for using the ROS framework for the development of algorithms facing a competitive challenge, e.g. exploration algorithms. The obtained students' opinions were instructive, reporting, for example, that although they consider harder to master ROS when compared to Matlab, it might be more useful in their (robotic related) professional careers, which enhanced their disposition to study it. They also considered that the challenge-exercises, in addition to motivate them, helped to develop their skills as engineers to a greater extent than the skeleton-code based ones. These and other conclusions will be useful in posterior courses to boost the interest and motivation of the students.

Keywords: Motivational Learning, Robotic Courses, Matlab, ROS.

1 INTRODUCTION

Robotics is a hot field with strong theoretical and practical backgrounds [1]. As a consequence of this, any course targeted at providing the principles behind robotics-related problems must support the theory lessons with well-designed practical exercises. Typically, the topics to be studied in robotics undergraduate courses include: robot motion, perception, localization, environment mapping, motion planning, and robotic control architectures among others.

Robotics courses require any software framework to complete the practical exercises. Even if robotic platforms are available, it is required a software framework to develop the algorithms that are subsequently loaded into the robot. At this point two options exist: the utilization of a general framework, like Matlab [2] or Octave [3], or the usage of a robotic-targeted one, as is the case of the Robot Operating System (ROS) [4], the Open Mobile Robot Architecture (OpenMORA) [5], or the Carnegie Mellon Navigation (CARMEN) Toolkit [6]. The advantage of general frameworks is that their utilization is not limited to robotic problems, so learning them can be profitable in a variety of domains. On the contrary, it is not common to see real robots running software generated by these frameworks, so their programs usually must be translated in order to be exploitable. Regarding robotic-targeted frameworks, they produce programs that are ready use by robotic platforms, although they normally involve a slow learning curve.

Once a certain framework is chosen, different pedagogical techniques can be followed in order to conduct the practical exercises. For example, if the exercises are about the motion of a robot using velocity commands, an option is to provide a skeleton of the code to simulate the robot motion with gaps to be filled by the students. These gaps are parts of the codified algorithm of particular interest. This approach makes the students to pay special attention to those parts, with the goal of reinforcing their understanding about them although, at the same time, limiting their creativity and motivation. An alternative approach to keep their motivation high is to set an exercise –without skeletons– where the developed algorithms have to face some type of challenge: in the case of the robot motion, for example, to give the most appropriated velocity commands so the robot stops as closer as possible to a certain location in a given time.

This paper presents the experiences on using two different learning strategies when designing the practical lessons of robotics, undergraduate courses. The study was carried out with the goal of enhancing the interest and motivation of the students in this particular field. Concretely, the first strategy employs Matlab for the filling of skeletons of code addressing typical robotic problems like robot motion, localization, or mapping, while the second one opts for the ROS framework for the development of algorithms facing a given challenge. The robotic courses were delivered by staff from the Machine Perception and Intelligent Robotics group (MAPIR) [7] in the University of Málaga, with an extensive experience in those fields [8][9][10][11]. The students taking the courses are from two different degrees: Computer Science and Industrial Engineering, having different skills and motivations, so it is interesting to measure the divergences in their opinions about the two resorted pedagogical strategies.

After completing the practical exercises, a number of questionnaires were proposed to the students, from where we got a valuable and instructive feedback for posterior editions of the courses. For example, the pedagogical strategy where the practical exercises are presented as challenges, without any skeleton of code, motivated more the students, so it could be interesting to include more exercises following that approach. The questionnaires also reported clear differences about the previous students' knowledge about Matlab depending on their degrees, as well as about their abilities for learning the concepts behind ROS, giving us directions for improving the students' experience in next courses.

2 MATERIAL AND METHODOLOGY

2.1 Software frameworks

In order to support the theoretical lessons, which are grouped in the following topics: robot motion, robot sensing, localization, mapping, SLAM, motion planning, and control architectures, two software frameworks are used: a mathematical-oriented (Matlab), and a robotic-oriented (ROS). The next sections briefly describe them.

2.1.1 Matlab

Matlab [2] comes from MAtrix LABoratory, and consists of a numerical computing environment accompanied by its own programming language. It is developed by MathWorks®, and provides easy-to-use APIs for matrix manipulations, visualization of data, design of user interfaces, or implementation of algorithms, which are interesting from the point of view of teaching. Matlab can be also interfaced (although not straightforwardly) with programs written in popular languages like C, C++, Java or Python.

Control engineering was the first field where Matlab gained in popularity, although thanks to its features it was quickly adopted in education (linear algebra, numerical analysis, image processing) and its usage is an increasing trend among robotics courses. This is mainly due to the proliferation of toolboxes giving the opportunity of showing the operation of different robotic algorithms – also spending a reduced time coding. Examples of these tools are:

- The Robotics System Toolbox™ [12] from Mathworks, which includes functionality for map representation, path planning, and path following for differential drive robots.
- Robotics Toolbox for Matlab [13], useful for the study and simulation of both, classical arm-type and mobile robots, offering path planning algorithms, kinodynamic planning, localization, map building, and simultaneous localization and mapping.
- ARTE: A Robotics Toolbox for Education [14], a toolbox for simulating robotic, industrial manipulators.

Given the commented features of this software framework and the available toolboxes, Matlab was used to complete some of our practical lessons, as commented in Section 2.2.

2.1.2 Robot Operating System

The Robot Operating System (ROS) [4] is a framework for developing software related to robotics. It consists of a number of tools, libraries, and conventions with the goal of easing the implementation of software for controlling different types of robots. It is built upon the idea that the development of the capabilities of an autonomous agent spans over many different issues that are difficult to be mastered at the same time, so the creation of an ecosystem with common software resources would permit to collaborate and join efforts for achieving such a goal. This collaboration is empowered with tools like a repository of software packages, a wiki (http://wiki.ros.org/), or a forum (ROS Answers, http://answers.ros.org/) among others.

ROS was designed to be distributed and modular, in such a way that any user can employ as much or as little of it as needed. Concretely, ROS includes the following core components:

- Communications infrastructure: a message passing interface to communicate the different software components of a robot at a low-level.
- Robot-specific features: ROS also provides common robot-specific tools including standard message definitions, the Robot Geometry Library, the Robot Description Language, and packages for diagnostics, pose estimation, localization, mapping, or navigation.
- Tools supporting introspecting, debugging, plotting, and visualizing the state of the system being developed. Examples of these tools are *rviz* (for the visualization of data from different sensors and robot models), *rqt* (for the development of graphical interfaces for a robot), or *rqt_bag* (for collecting and playing back data).

Nowadays, ROS is the de-facto software in most robotic platforms, so we have relied on it for conducting some of the practical lessons, as described in the next section.

2.2 Pedagogical techniques

Given the particular features of the two software frameworks used, two different pedagogical techniques have been followed when presenting the practical exercises to the students: (i) skeletons of code from a Matlab toolbox with gaps to be filled (Section 2.2.1), and (ii) less guided exercises within the ROS framework where the developed algorithms have to face some type of challenge (Section 2.2.2).

2.2.1 Skeleton code approach

For most of the exercises using Matlab we resorted to the code provided by Paul Newman (BP Professor of Information Engineering at the University of Oxford) in its popular set of lectures about navigating mobile robots[1]. Parts of this code were removed, containing relevant concepts which understanding by the students is especially important in the course. Once correctly filled, the algorithms must be able to deal with the following robotic issues:

- Robot motion: control the differential motion of a robot through velocity or odometry commands (see Fig. 1).
- Localization: localize a robot endowed with a range sensor in a map with respect to a set of landmarks employing Least Squares Global Localization, or supposing a robot equipped with a range-bearing sensor and using the Extended Kalman Filter (see Fig. 2-left).
- Mapping: in this case the students have to localize the landmarks in the robot workspace (create a map of it) through an Extended Kalman Filter and a range-bearing sensor.

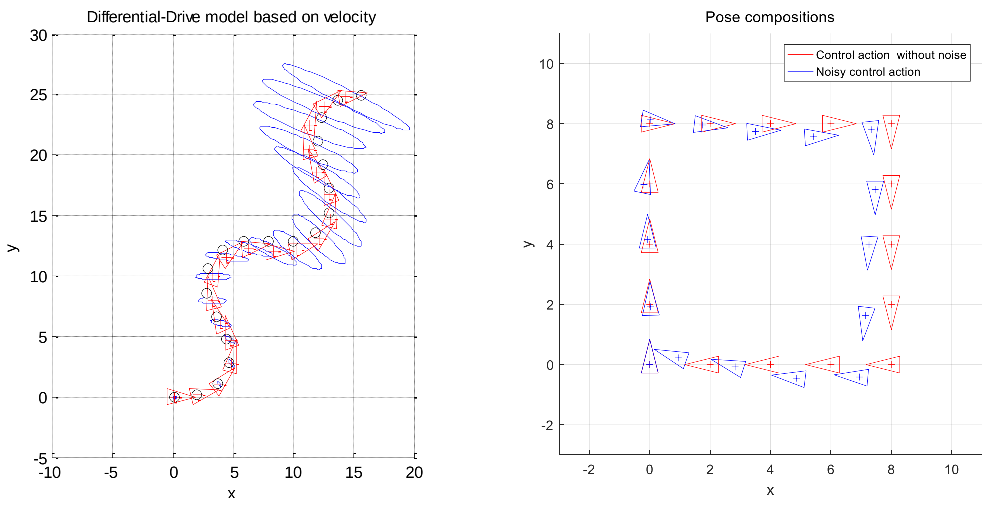

Fig. 1. Left, path of a robot moved through velocity commands without considering noise in the control actions (in red), samples of the path considering noise (black circles), and uncertainty about the robot position (blue ovals). Right, plot showing the path followed by a robot according to a number of control actions with and without noise employing odometry commands.

[1] Available at: http://www.robots.ox.ac.uk/~pnewman/Teaching/C4CourseResources/C4BMobileRobots.pdf

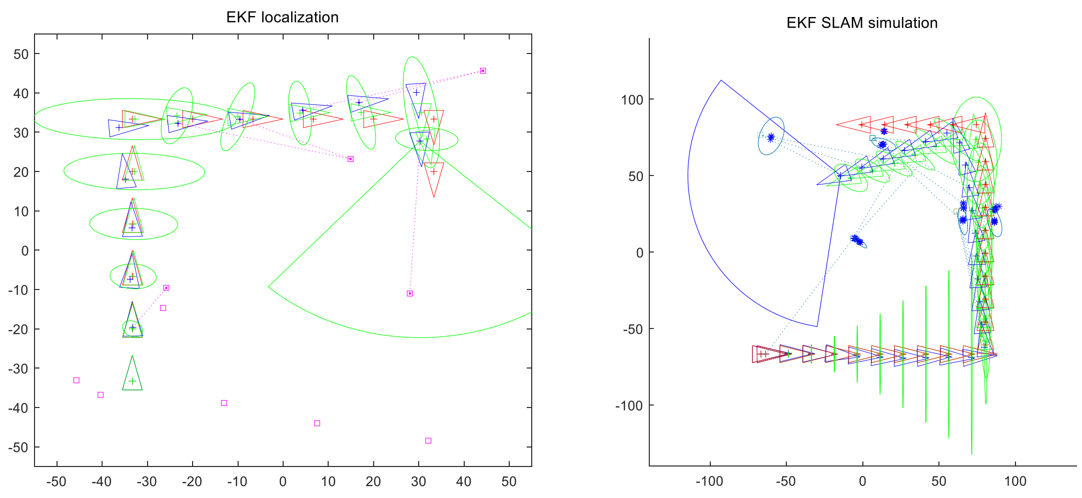

Fig. 2. Left, example of the localization process of a robot according to a number of landmarks (pink squares) using an Extended Kalman Filter. The red triangles stand for the path ideally followed by the robot, the blue ones are the real trajectory due to noisy control actions, and green ones represent the estimated one. Green ovals are the uncertainty about the robot location. Right, result of an SLAM algorithm. In this case the location of the landmarks is also being estimated.

- SLAM: simultaneously localize and build a map employing an Extended Kalman Filter and the same sensor (see Fig. 2-right).
- Motion planning: implementation of a reactive navigation algorithm based on Potential Fields, so the robot must avoid a number of obstacles between a starting and a goal position.

Thereby, for example, in the exercise where the students had to implement the reactive navigation algorithm, the skeleton of such algorithm was presented beforehand, and they just had to fill the parts where the repulsive, attractive, and total forces of the Potential Fields approach are computed. This has the clear advantage of saving time to the students, and permits the lecturer to focus on the part of the algorithm that s/he considers more important to review. However, it is also clear that the understanding of the full algorithm can be compromised, and that the motivation of the student when it faces several exercises following this pedagogical technique can be affected.

We have made available some exercises following this skeleton-based approach, also containing the skeleton codes, as well as other resources of the courses[2].

2.2.2 Motivational approach

The second pedagogical technique pursues the enhancement of the motivation of the students by means of challenges. For that, we designed an exercise within the ROS framework where the students had to implement an *explorer robot*, that is, to develop an algorithm to make a robot to explore an unknown environment, covering it as much as possible in a given time. An algorithm like this has many applications: map building, rescue tasks, security works, etc.

To save time to the students, we provided a *virtual machine* (Virtual Box) with Xubuntu 14.04 OS, and ROS Indigo installed and ready to use. The students could check the success of their algorithms employing the already installed Stage simulator[3] (its wrapper for ROS) [15] and any of the 5 environments that were released for that purpose: 4 from home environments, and another one from an office setting. Fig. 3-top shows two of these environments. The students were also told that their algorithms were to be tested in 5 additional environments, just to check the generality of their solutions.

All the students had to use the same robot, endowed with a laser scanner as a sensor to perceive the environment and a robotic base yielding odometry information. They also must move the robot employing velocity commands and avoiding collisions with walls. Some of the explorations done by the students' algorithms can be seen in Fig. 3-bottom.

[2] Available at: http://mapir.uma.es/mapirwebsite/index.php/papers/257

[3] Stage provides a virtual world populated by mobile robots and sensors, with objects for the robots to sense and manipulate.

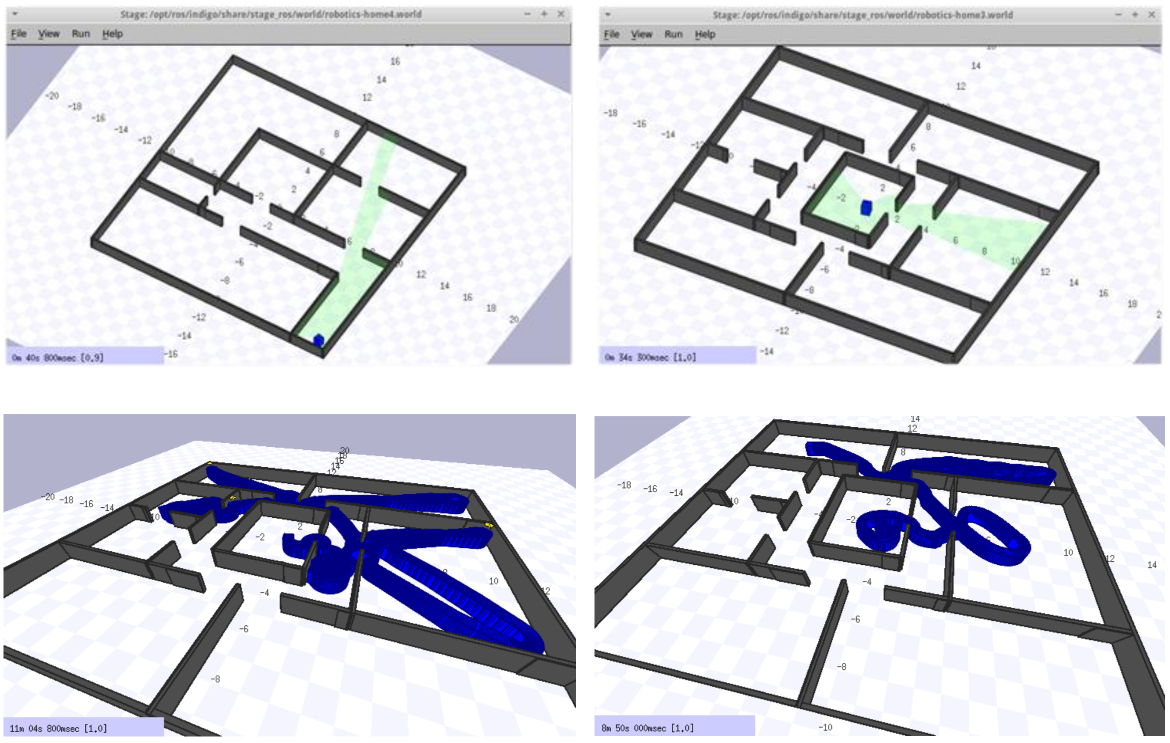
Stage: /opt/ros/indigo/share/stage_ros/world/robotics-home4.world
File View Run Help
Stage: /opt/ros/indigo/share/stage_ros/world/robotics-home3.world
File View Run Help

Fig. 3. Top, two of the home environments provided to the students: 1 office (bottom-left one) and three homes. The blue box represents the starting location of the robot, while the green shape is the field of view of the sensor. Bottom, two examples of the exploration of the robot performed by the algorithms of two of our students.

2.3 Data acquisition

After completing the practical lessons, the students: 11 in Computer Science, and 12 in Industrial Engineering, were proposed to fill an electronic questionnaire with questions related to: the experience, ROS/Matlab, the exercises completed, and the knowledge acquired. The obtained feedback was analyzed with 2016 Microsoft Excel and Matlab, yielding relevant information about the opinions of the students according to both: their degrees, and in general. The extracted results are elucidated in the next section.

3 RESULTS

3.1 Experience with ROS

The first group of questions was related to the experience of the students with the ROS framework during the lessons. Fig. 4 reports the students' feedback, showing interesting differences among the students from both degrees. The students from Computer Science (CS) found it easier (on average) to learn the core concepts of ROS (ROS master, topics, nodes, etc.), as well as how to implement ROS programs, when compared to the students from Industrial Engineering (IE). They also felt more comfortable programming ROS code, as it is shown by the difference of almost one point on average (from a scale of 5) in their answers (Q3). This can be due to the specific skills acquired during the previous degrees' courses, being the CS students used to developing algorithms in different IDEs, frameworks and programming languages. However, all the students agree regarding the utility of ROS for programming robots (Q4).

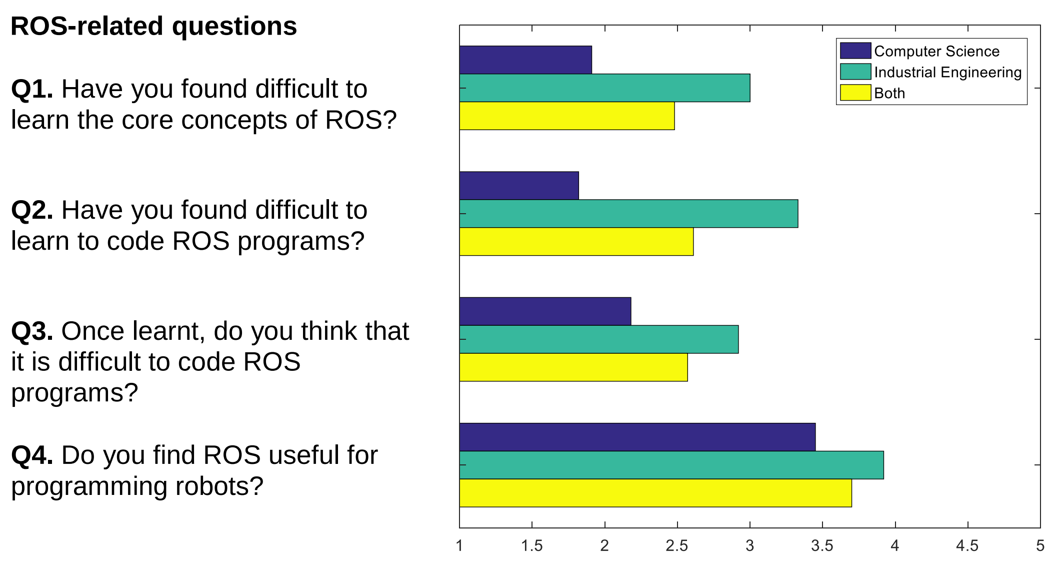

Fig. 4. Results obtained from the ROS-related questions according to the students' degrees. In the 1—5 scale, 1 means no/disagree and 5 yes/agree.

3.2 Feedback about the pedagogical techniques

This set of questions is focused on obtaining the opinion of the students about the two pedagogical techniques employed in the experience. Fig. 5 shows the obtained results, where we can see that in this case the students from both degrees mostly share their opinions. The first question is the one with the most affirmative feedback in the questionnaires (Q5): the students found the motivational exercise with the exploration challenge positive (almost 4 points on average). They also felt more motivated when facing the exercise than dealing with those following the skeleton-based technique, as it is reported by Q6. The students were also asked about the development of their skills as engineers when completing the exploration exercise (search of innovative solutions, programming abilities, etc.), obtaining a positive feedback (Q7). They also considered that it did to a greater extent than the exercises based on filling skeleton codes (Q8). Finally, it is interesting to comment the gathered results about Q9. In this question

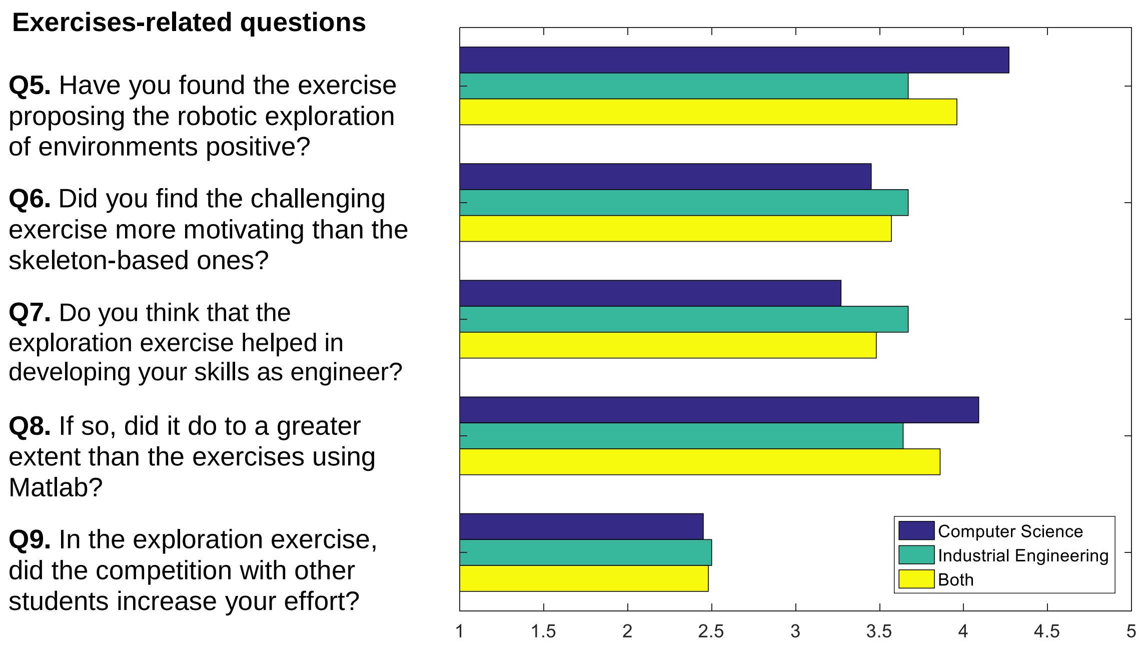

Fig. 5. Feedback obtained from the students concerning their opinion about the completed exercises. In the 1—5 scale, 1 means no/disagree and 5 yes/agree.

students were asked if the competition set in the exploration exercise, where the performance of their developed algorithms would be compared and made public (only for the students in the courses), made them to increase their efforts, obtaining a modest result of 2.5 points.

3.3 Acquired knowledge

The last set of questions seeks to obtain the students' opinions about their acquired knowledge (see Fig. 6). The first question (Q10) had an affirmative answer (3.9 points): the students considered that the concepts learnt about ROS could be useful in the future if they chose a career related to robotics. They also pointed out that, in such a case, those concepts would be more relevant than the ones learned about Matlab (Q11). This makes sense since, as commented, ROS is a framework focused on robotics applications while Matlab is more general.

A significant difference appears in Q12, which measures the opinion about the students' knowledge of Matlab previous to the courses (3.2 points from IE students vs. the 2.2 points from CS ones). This is due to the utilization of Matlab in a higher number of subjects in the IE degree. Nevertheless, their knowledge after the course was, in their opinion, the same (Q13). The results obtained concerning the previous knowledge about ROS got the lowest punctuation in the questionnaires (1.3 points), which makes sense since it is only related to robotics courses. However, the students considered that at the end of the courses they achieved suitable skills on the ROS framework.

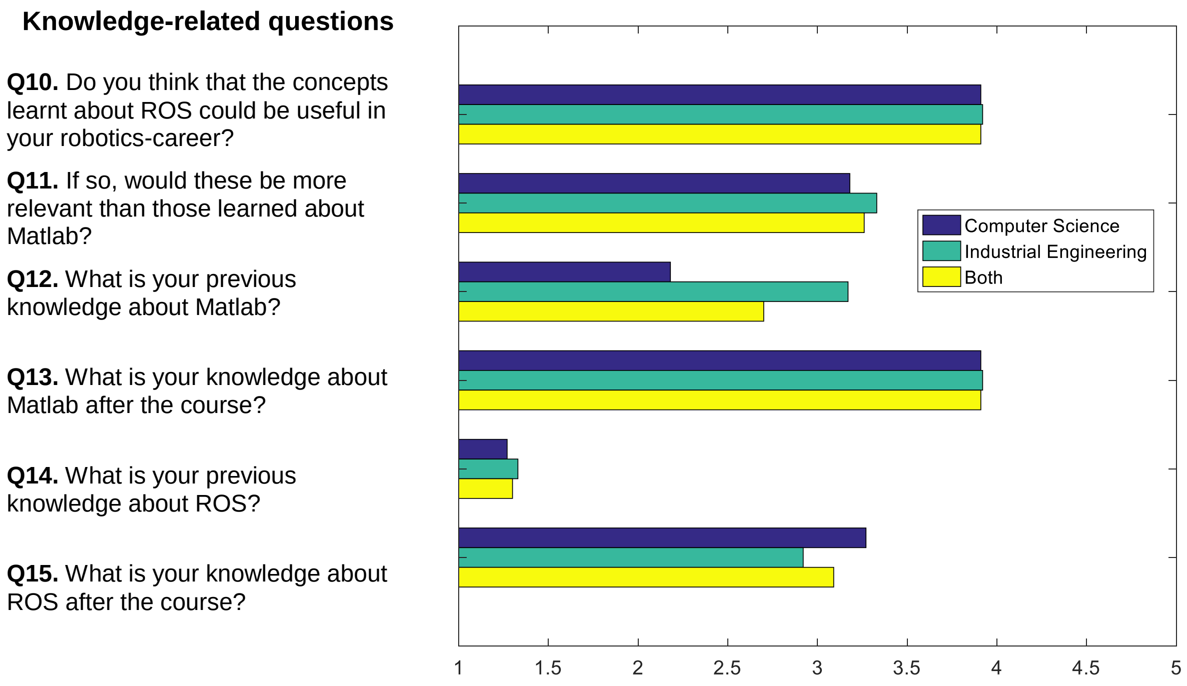

Fig. 6. Feedback from the students about the knowledge acquired during the courses. In the 1—5 scale, 1 means no/disagree/no experience and 5 yes/agree/master the framework.

4 CONCLUSIONS

This paper has presented our experiences with two learning approaches based on different designs of practical exercises targeted at undergraduate robotics courses. The first approach provided the students with skeletons of code with gaps to be filled, implementing a number of algorithms facing typical robotic issues: robot localization, mapping, robot motion, path planning, etc. Such algorithms had to be developed within the MATLAB software, a well-known framework where a number of robotics toolboxes are available. Contrary, the second approach aimed to motivate students with the development of an algorithm facing some type of challenge. Concretely, students had to implement a robotic explorer that must visit an area as large as possible of an unknown environment in a given time. In this case the algorithm had to be developed within the ROS framework, which is oriented to robotic applications. To save time to the students, we also provided them with a virtual machine holding a Xubuntu 14.04 OS and a ready-to-use ROS installation.

To evaluate the motivation of these students when facing practical lessons from the two approaches, we proposed them three questionnaires at the end of the courses. The analysis of the students' feedback provided valuable directions for enhancing the motivation of the students in posterior editions of the courses, factor that we consider a good indicator of the learning success. For example, they gave us a favourable opinion about the motivational learning approach: they found positive the experience and considered that helped to develop their skills as engineers to a greater extent that completing skeleton-based exercises. Furthermore, given that the same pedagogical strategies were followed in courses from two different degrees: Computer Science and Industrial Engineering, we also obtained instructive information about their previous knowledge, as well as about their capabilities to learn the notions of the tools employed in the practical lessons. In this way, aiming to enhance their experience in the courses, future lessons could be adapted according to the characteristics of the students in each degree.

REFERENCES

[1] Thrun, S., Burgard, W., and Fox, D. (2005). Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press.

[2] Matlab. http://es.mathworks.com

[3] Eaton, J.W., Bateman, D., Hauberg, S., and Wehbring, R. (2014). GNU Octave version 3.8.1 manual: a high-level interactive language for numerical computations. CreateSpace Independent Publishing Platform, ISBN: 1441413006.

[4] Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software.

[5] Blanco, J., Galindo, C. Monroy, J. G., and Gonzalez-Jimenez, J. (2010). Open Mobile Robot Architecture (OpenMORA). [Online]. Available: http://www.mapir.isa.uma.es/openmora

[6] Montemerlo, M., Roy, N., and Thrun, S. Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) Toolkit. In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, Nevada, Oct. 2003, pp. 2436–2441.

[7] Machine Perception and Intelligent Robotics group (MAPIR) webpage: http://mapir.uma.es

[8] Ruiz-Sarmiento, J.R., Galindo, C., and González-Jiménez, J. (2016). Building Multiversal Semantic Maps for Mobile Robot Operation. International Journal of Knowledge-Based Systems.

[9] Ruiz-Sarmiento, J.R., Galindo, C., and González-Jiménez, J. (2015). Joint categorization of objects and rooms for mobile robots. International Conference on Intelligent Robots and Systems (IROS).

[10] González-Jiménez, J., Ruiz-Sarmiento, J.R., and Galindo, C. (2013). Improving 2D Reactive Navigators with Kinect. 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Reykjavic, Iceland.

[11] Cruz-Martín, A., Fernández-Madrigal, J.A., Galindo, C., González-Jiménez, J., Stockmans-Daou, C., Blanco-Claraco, J.L. (2012). A Lego Mindstorms Nxt approach for Teaching At Data acquisition, Control systems engineering and Real-Time systems undergraduate courses. Computers & Education, Vol. 53, Issue 3.

[12] Robotics System Toolbox webpage: https://mathworks.com/products/robotics.html

[13] Corke, P.I. (2011). Robotics, Vision & Control. Springer, ISBN: 978-3-642-20143-1.

[14] ARTE: A Robotics Toolbox for Education webpage: http://arvc.umh.es/arte/

[15] The Stage Robot Simulator webpage: http://playerstage.sourceforge.net/doc/Stage-3.2.1/