Innovating in robotics education with Gazebo simulator and JdeRobot framework

José M. Cañas[1], Laura Martín [1], Julio Vega [1]

[1]*ETS Ing. Telecommunication, Universidad Rey Juan Carlos, Camino del Molino s/n, Fuenlabrada, +34 91 488 87 55, josemaria.plaza@urjc.es*

Abstract. Robotics is becoming an interesting subject because of its technological appeal to students and because it is a good way to practice many concepts of engineering, mathematics, programming, physics, etc. .. In this paper we present the design and content of one course in the field of robotics which use innovative tools for practice: Gazebo simulator and JdeRobot software framework. Gazebo is an open source 3D simulator very extended for research purposes, the reference in ROS standard platform, and recently chosen by DARPA for international DARPA Robotics Challenge (DRC). Using a realistic simulation, it allows students to learn with robots and expensive devices (humanoids, lasers, etc.), using standardized research tools. JdeRobot is an open source and component oriented middleware for creating robotics, computer vision and home automation applications. In the designed education course the access to sensors and actuators has been simplified allowing the student to focus on programming the system intelligence. The communications middleware and GUI management are hidden. Students programs, without any change, can be tested both on simulated and on available real robots. This robotics course has been experimentally validated in the last three years with master and degree students in engineering and communications, getting good feedback from them.

Keywords: Education, robotics, simulators.

1. INTRODUCTION

In the last years many advanced countries like US, Germany and Japan have detected a decrease of students in engineering and technological degrees. The students perceive them as hard and difficult degrees with not very well paid job offers in comparison with other ones. As part of the United States government's goal of preparing students for a STEM-based economy, the Obama administration has committed $3.1 billion to improve STEM education nationwide.

Robotics is appealing for students and a good scenario where they can practice technology, mathematics and science concepts (STEM). Many robotic competition like First Lego League or Hispabot have been fostered to gather the students enthusiasm and approach young students to technology in a funny way. Robots have also been used in secondary education, for instance the “Technology” course at several High Schools in Spain.

In addition, robotics is a rapidly expanding field of engineering and its popularity has increased astonishingly in the last few years. Maybe as new robotic products, applications and prototypes (like the Roomba cleaning robot, the Google Car, robot arms in factories and drones to name a few) have reached the global market. Currently, there is strong industry demand for computer vision and robotics engineers and scientists. The required profiles are people who understand computer vision or robotics technology and know how to apply them in the real world. Learning these topics include habilities such as programming, image processing, calculus, linear algebra, numerical methods, etc.

Robotics is an application domain of different technologies and knowledge areas, and it can be seen from different perspectives. First, one traditional view is inside Electronics and Electrical Engineering. Teaching robotics from this point of view makes emphasis in the robot building, its mechanical parts, sensor devices, motors, electronic design, processors and its basic programming. Second, robotics can be also considered a field inside Computer Science (CS) and then the main focus is programming the robot. Once the robot is already built its intelligence lies on its software. Perception, navigation techniques or decision making algorithms are all finally implemented in software. Good algorithms and programming provide better robot behavior and functionality.

Simulators are one common tool for robotics engineers. They allow testing and debugging the software on virtual environments before testing it on the real robots, and simulate real processes to gain better and faster understanding of the underlying principles. Some simulators only support 2D worlds, others fully support 3D worlds, complex sensors like cameras, depth sensors, etc. and different robots geometries and platforms. Besides their use by real engineers in research and industry, the simulators open new possibilities in teaching robotics.

According to the 'White Book on robotics in Spain' (GTRob, 2011) no university in Spain offers a robotics specific bachelor program, but several of them provide robotics master programs. Most of the times there are isolated robotics courses inside technological masters or bachelor programs, and that is the case of Rey Juan Carlos University for which we have developed the robotics course design and practice environment described in this paper.

In next section some related works in the field of teaching robotics are presented. Third section briefly reports the chosen simulator and the software framework we have developed for robotic practice. Fourth section describes the syllabus of the master course and the particular robotics exercises we have designed. Some conclusions are summarized in the final section.

2. RELATED WORKS

Small robots and small hardware platforms have been used in teaching robotics since late 90s (Candelas et al., 2006). RugWarrior (Jones et al., 1998), EyeBot (Braunl, 2007), Lego robots (Galvan et al., 2006) are illustrative examples. However, the experience have shown that this kind of platform is more useful to teach digital electronic or microcontroller design instead of mobile robotics. For instance, Lego Mindstorms provide very limited possibilities to implement sophisticated algorithms, because this robot have very simple sensors and little computing capacity. Anyway this platform has been used to introduce robotics in pre-universitary courses (Balch, 2008) and graduate levels (Menegatti y Moro, 2010).

Another interesting academic focus is the robotics teaching based on projects. It is necessary to set the objective course, for example, to participate in a robotic competition and during the process of the robot construction is possible to teach the student several practical knowledge. One example of this method is Xavier, created by the students of the Carnegie Melon University (EE.UU.) under the supervision of Reid Simmons with the objetive to participate in the AAAI Robotics Competition in 1993. Other example more recent is the robot Stanley winner of the Grand Challenge in 2005, created by Stanford students under supervision of Sebastian Thurn. However this methodology is possible to apply in excellence centers with few students and many resources.

Today there are also several MOOC’s (Massive Open Online Courses) as a way to learn robotics or artificial vision. MOOC’s became popular in early 2012 when Daphne Koller and Andrew Ng launched their on-line learning platform Coursera. The main providers of MOOC's are Coursera, Udacity and edX. The first artificial intelligence course was spectacularly successful, especially the first with 160,000 students enrolled. This courses offer accessible, affordable, engaging classes that anyone can take, any time, providing video lectures, practices and exams. This kind of courses show the importance of simulation because it is impossible to give one robot to all the students. Using simulator the students could put into practice the knowledge acquired in the theoretical classes.

The robotics engineering program in the Worcester Polytechnic Institute uses real robot to develop the practices (Padir et al., 2011). The students work in teams on all project assignments. Incorporating open-ended projects with detailed timelines and milestones. At the University of Minnesota, computer science is taught through robotics projects, and computer vision is used as a navigational tool (Gini, 1996). Manufacturing has also been used to teach robot vision (Liang, 1996). At Carnegie Mellon University robotics is also used as teaching tool (Krotkov, Feb 1996). In the last years Pyro framework (Blank et al., 2006), an open-source Python robotics toolkit, is used there for exploring topics in AI and robotics.

3. SOFTWARE TOOLS FOR TEACHING ROBOTICS

In this section the software tools created or used for the students to develop their practices are described. First, the Gazebo simulator and the JdeRobot framework.

3.1. Gazebo simulator and reference robot: Pioneer 2DX

Gazebo[1] is the preferred simulator in JdeRobot framework. It is a 3D open source simulator which offers a rich environment to quickly test multirobot systems and which simulates several robots and cameras in a realistic way. All simulated object have mass, velocity, frictions and numerous other attributes that allow them to behave realistically when pulled, knocked over or pushed.

The reference robot for the practice of the designed course is the Pioneer 2-DX model made by ActivMedia Robotics. The students have to program its intelligence. It is an autonomous wheeled robot used in many projects due to its versatility and robustness. The robot's motion is achieved by two driving wheels and a thirds one that

[1] http://gazebosim.org

moves freely with the movement of the robot, the ease of manoeuvring, together with the robot's reduced size (50x49x63cm), implies a very good choice for indoor environments navigation. The robot's default configuration is illustrated in Figure 1, both the real pioneer and the simulated pioneer configurated with encoders, laser, stereo cameras, sonars and motors

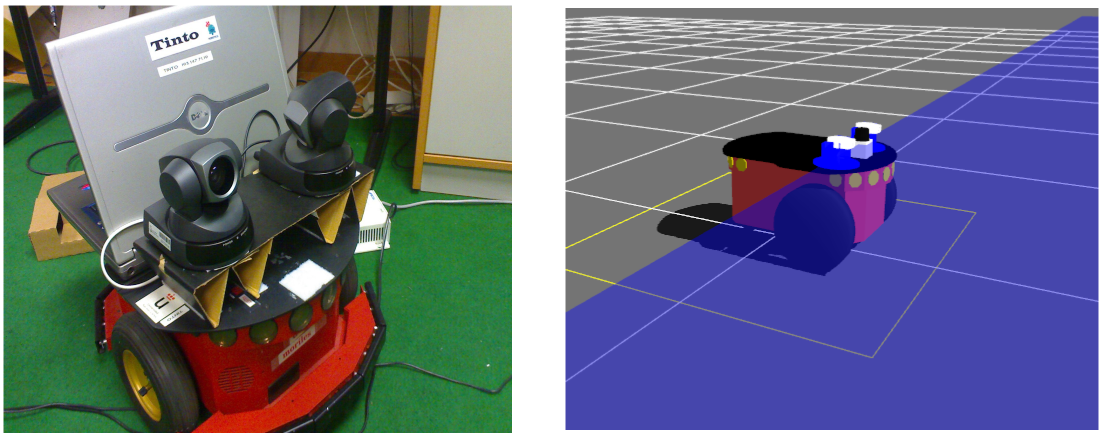

Figure 1. (a) Real pioneer (b) Pioneer simulated on Gazebo

3.2. JdeRobot

The software tool used to develop the student practices has been created inside the JdeRobot[2] framework (Cañas et al., 2013). It is an open source software framework with the philosophy of providing an easy method for creating robotics, computer vision and home automation applications. It provides a component-based programming environment where the applications are compound of a collection of several concurrent asynchronous components. They perform simple and specific tasks, and interact among them. The concurrent execution of multiple component results in a behaviour. Besides, JdeRobot simplifies the access to actuators of getting sensor measurement by simply write or read form a local variable. It also makes easy the reuse of previous implemented applications or components.

JdeRobot uses ICE as communications middleware between these components, which can be written in different programming languages (C++, Python , Java...) and run on distributed machines. JdeRobot doesn't work alone, it use several external

[2] http://jderobot.org

libraries to extend the functionality such as simulators (Gazebo or Player/Stage), OpenCV, ICE, PCL, OpenNi, etc.

JdeRobot includes several driver-components that communicate with the different devices, sensors or actuators. Encoders, cameras, laser, Kinect or pan & tilt and different robots like pioneer 2-DX, Kobuki, Ar.Drone or Nao are supported. For the support of simulated robots several Gazebo plugins have been developed (Figure 2), which provide the same ICE interface than the homologous driver component for the corresponding real device.

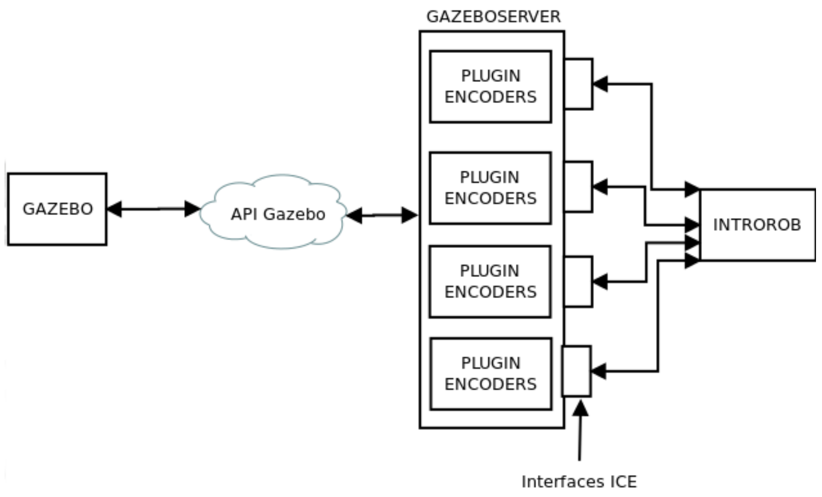

Figure 2. JdeRobot plugins for Gazebo to manage simulated sensors and actuators.

JdeRobot also includes some useful tools and libraries. *Progeo* is a projective geometry library and used to relate, in both senses, 2D visual information and 3D spatial information. *Progeo* uses a pinhole camera model. It provides the backproject function (to obtain the projection line that connect the camera with the focus and the 3D ray that projects in a pixel of the image plane) and the Project function (to project a 3D point of the world to the corresponding 2D pixel of the camera image).

3.3. Introrob component

Regular students don't have too much time to learn all the issues and power behind JdeRobot. They just want to use it as soon as possible focusing on their robotic practices. To hide most the complexity of robot programming an let the alumni to focus on the key aspect of robot control (not in GUI neither in communication middleware etc) a component named introrob has been developed. This tool is used to teach robotics in the MSc courses. This component has been developed for students to simplify the development of robot control applications in their practices (algorithm oriented to object recognition, navigation algorithm, autonomous behaviour, etc).

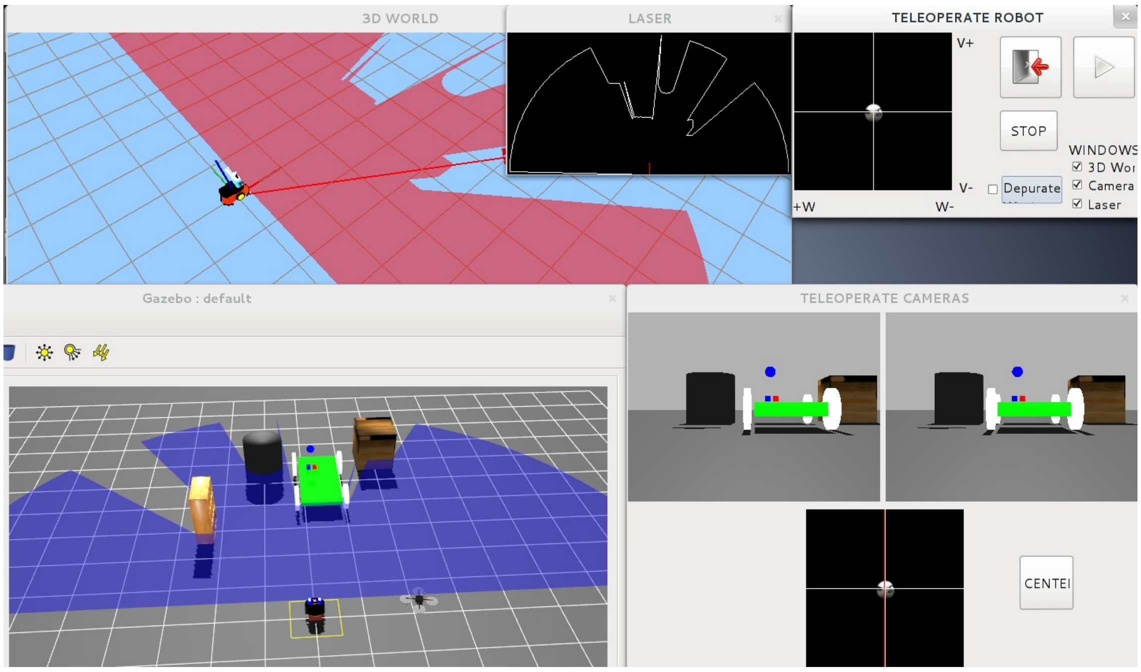

Figure 3. Introrob Graphical Interface

Figure 3 shows the user interface. The main window at the top right contains a joystick to control the robot, three check boxes that open three other windows, a button to stop the robot and other button to start or stop the student algorithm. Top area displays the laser's measurement and a OpenGL world that contains a representations of the robot and the sensors. The bottom-left area shows the Gazebo simulated world and in the bottom-right area the onboard camera images are displayed.

As shown in Figure 4 Introrob provides a very simple local API in C++ with functions to get the sensors reading and set the motors commands. For example, if the students need the laser measurements vector they should call the *getLaserData function*. To access camera information the functions *getImageRight* or *getImageLeft* provide the image data as a OpenCV structure (cv::Mat).

In addition, Introrob provides a template for a robot control application divided in two parts. One of this part is the control thread running periodically (once overy 100ms), this thread requests the sensors information, sends the data to the actuators and executes the algorithm developed by the students. And the other part is the user interface thread that show the processing images, robot information or a 3D visualizer. This template is provided in the *MyAlgorithm.cpp* file which the students are required to modify to embed their control software.

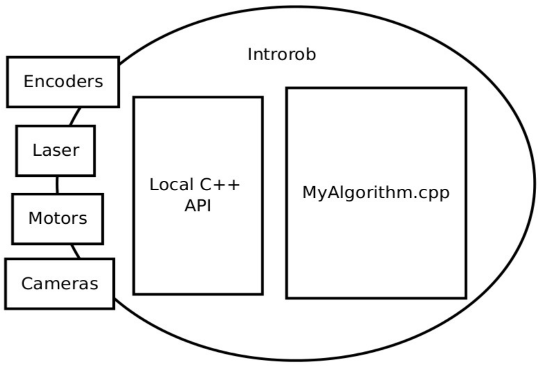

Figure 4. Introrob's template block diagram

4. MASTER COURSE DESIGN

The proposed course is divided in twelve weeks, 8 to explain the theoretical concepts and 4 to perform the practices. The syllabus of *Robotics* course first (1) introduces students to the state of the art in the field of mobile robotics. Then presents the essential components of any robot: (2) sensors and (3) actuators). The fundamentals of reactive control architectures are explained in the (4) lesson. Following units focus on the basic problems of autonomous robotics and the most successful techniques to resolve them: (5) local navigation (6) global navigation (7) mapping and (8) localization. Finally (9) robot architectures and (10) vision in robotics are introduced.

The ability to program a mobile robot for different tasks is a very important part in the course because the intelligence of the robot resides in its software. Two practices are proposed.

4.1. Visual control practice

The first practice is related with the visual control lesson and requires programming the robot to follow the blue line drawn in the floor shown in Figure 6.1. Figure 5 shows the practice as a black box. The only input to the algorithm are the left camera images. The output is the velocity and the angular velocity commaned to the robot motors.

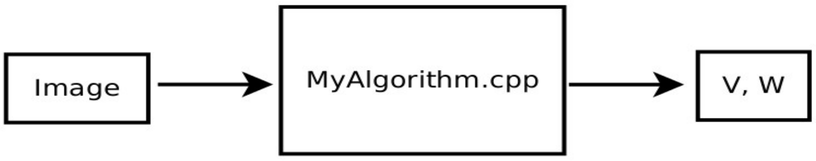

Figure 5. Block diagram of visual control practice

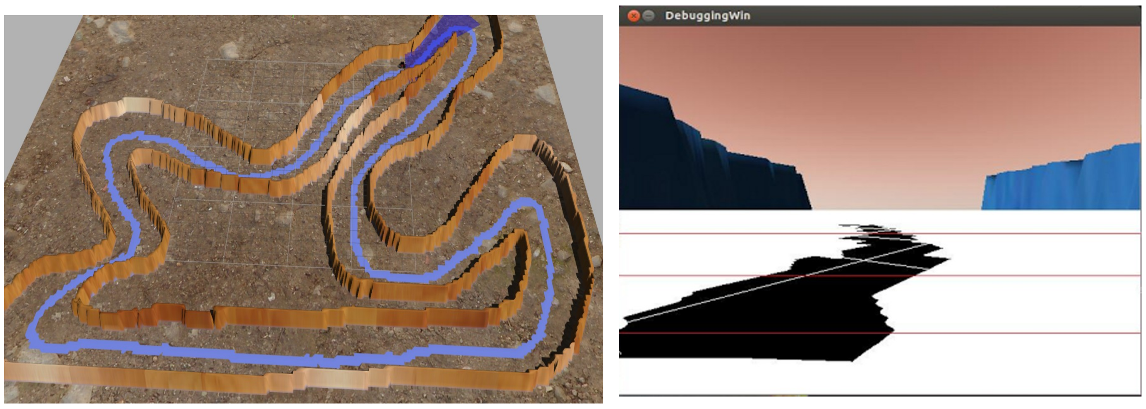

Figure 6. (a) Follow the line world (b) Processing image from a student.

In this practice the students have to put into practice several knowledge received about visual control, image processing or programming skills. The practice was evaluated measuring the time required to complete a lap in a given simulated circuit and competing with the solutions of others students.

A typical solution of this practice is using a PID controller and a color filter. The first step is tune the color filter in the color of the line, in this case blue (Figure 6(b) shows a processed image by a student). The second step is to find the edge of the blue line in several image rows. The error for the PID controller is calculated as the difference between the center of the line in each row and the center of the image. The formula for the PID controller is equation (Eq.1), where the error is calculated as explained.

$$u(t) = K_p e(t) + K_d \frac{de}{dt} + k_i \int_0^t e(t)dt \qquad (1)$$

In addition, a finite state machine or a case based control can be included depending on the image information. It is possible also to use different PID controllers for straight and curves to improve the robot behaviour.

4.2 Local navigation practice

Obstacle avoidance is one of the key issues to successful applications of mobile robots systems. It combines the need of circumventing obstacles and the need to proceed towards the navigation target. The second practice proposed is the programming of the Virtual Force Field (VFF) algorithm for local Navigation. Figure 7 shows the practice as a black box with robot's encoders, laser measurements and the targets input and both the angular and linear commanded speeds as output.

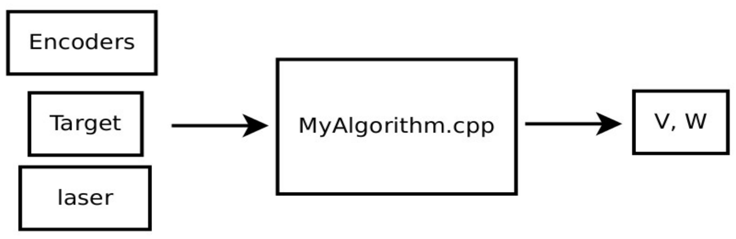

Figure 7. Block diagram of local navigation practice

Virtual Force Field technique generates two forces: the attraction force, generated by the target, and the repulsion, generated by the obstacles perceived with the laser sensor. The total force is the combination sof both forces following Equation (2). This algorithm allows to navigate avoiding obstacles while the robot approaches to the target. In introrob is possible to indicate the target clicking in the user interface.

$$\vec{F_t} = \alpha \vec{F_{rep}} + \beta \vec{F_{atr}} \tag{2}$$

Figure 8 shows the simulated world where the robot has to navigate. It contains a lot rooms and big corridors to put into practice local and global navigation. It is possible to see the three vectors of the VFF algorithm in the Introrob GUI: the repulsion force vector in red, the attractive force vector in blue and the total one in green.

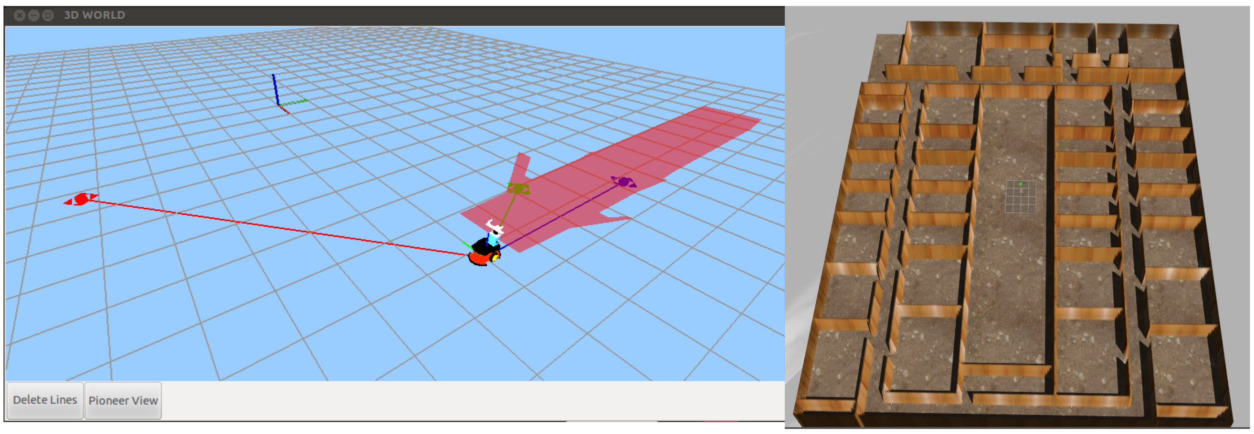

Figure 8. Target specification in 3D window and Navigation world

5. CONCLUSIONS

We have presented the design of one robotics course inside a Computer Science academic environment. For the practical side of the course we chose the Gazebo simulator and the JdeRobot middleware, and we developed an academic component, Introrob, for the student practices. We also designed several exercises including one on vision-based control, another on local navigation algorithm. This platform has been used for three years, with around 60 different students. The surveys at the end of the semester show that the students were glad with this environment for their practices.

The main conclusion of our robotics teaching experience is that powerful simulators like Gazebo are a good platform to learn robotics. It does not provide some lessons and details that lie in the use of real sensors and robots, but let us focus on the key aspects of the algorithms and robotic techniques, reducing the development time to have prototype software implementations. This way the students may face more practices in the same course.

The debate about using simulators is not new. They can not replace the experience with real robots. Typical disadvantages are their lack of realism, the legal problems or high price of licences, and the lack of generality (the students learn to use a simulator platform that maybe are not going to use in their professional lives). The realism achieved in the latest simulators and their physics engines is closer to true behavior of noisy sensors and actuators than ever, and good enough to realistic simulations. They support complex sensors like cameras and laser.

Gazebo is open source and free, so there is no problem at all with its price or intellectual property rights. In addition, it is becoming the de facto standard, with growing usage in robotics research community and industry world wide. And so, the gap between academic and professional tools is reduced.

Moreover, we have found additional advantages of using simulator for teaching robotics. First, sometimes we do not have a set of real robots to share among students, or the availabe robot is too expensive or too big to be portable. Each student may install the simulator on her computer and even work on it at home.

Second, they allow the creation of practices of increasing complexity, from a very basic setup to a very complex one, close to the state of the art research problems. Teachers can adapt the difficulty to the level of the students (bachelor, master, PhD).

Third, they extend the span of possible practices. For instance, students can focus on a global navigation practice assuming that self-localization is working fine, the simulator provides ground truth localization. This could not be possible with real robots, where the localization must be already solved before facing the global navigation.

Fourth, simulators simplify the visualization and debugging tasks. For instance, small real robots like the LEGO NXT, the EyeBot or Nao humanoid do not have any display or have a limited one and the debugging is typically done with sounds or lights. Using Gazebo the students' software itselft may open debugging windows.

The second conclusion is that software infrastructure for robotics education must hide many of the underlying details and complexity of robot programming, simplifying it to let the student focus on the algorithmic part of her practice. The presented course takes 14 weeks and dedicating part of them to show all the complexity of the robot software may distract students from the crux of the problems, and reduce the number of issues described along the course. Details are necessary but they are not the target of the classes. Our Introrob component hides the underlying details of getting sensor readings or sending motor commands through ICE messages, they both appear to the student as simple local function calls. Introrob also offers Graphical User Interface for debugging and a template for iterative execution, so students only have to insert their code into the template. Using Introrob they focus on the robotic crux of the practices.

The third conclusion is that installation of the environment must be easy. This is a difficult issue as Gazebo and JdeRobot are complex, have many dependencies and the target computers are very heterogenous as each student has her own machine and setup. Some years ago we provided tarballs with the source code and suggested several typical steps for compilation and installation (with autotools). This approach was painful as most of our students have limited computer science habilities and there were many machine dependent details. So they took too much time just to have the robotic framework installed on their computers. To reduce this set up time we prepared a set of debian packages to install all the software, solving there all the dependencies and relying on the existing packages (like the Gazebo for Ubuntu users).

As future lines we are thinking on two new robot platforms: the TurtleBot and the Nao humanoid. We have now both real robots in the lab and we are working on the Nao support in Gazebo. The idea is to offer the best students the chance to test their code on a real robot as a reward for their effort. We are also planning new practices like vision-based self-localization algorithms and map building algorithms. In additon, we have developed some JdeRobot components in Python to explore the future use of this language to program the robots.

BIBLIOGRAPHY

1. TUCKER BALCH. Designing personal robots for education: Hardware and software and and curriculum. IEEE Pervasive Computing, 2008.

2. DOUGLAS BLANCK, Deepak Kumar, Lisa Meeden, y Holly Yanco. Pyro: A python-based versatile programming environment for teaching robotics. Journal on Educational Resources in Computing, 2003.

3. DOUGLAS BLANCK, Deepak Kumar, Lisa Meeden, y Holly Yanco. The pyro toolkit for ai and robotics. AI Magazine Volume 27 Number 1, 2006.

4. THOMAS BRAUNL. Robotics education using embedded systems and

simulations. AAAI Spring Symposium Robots and Robot Venues: Resources for AI Education, 2007.

5. J.M CAÑAS, M. GONZÁLEZ. González, A. Hernández, F. Rivas. Recent advances in the JdeRobot framework for robot programming. Proceedings of the 12th RoboCity2030 Workshop, Robótica Cognitiva, 2013.
6. CANDELAS, F., Torres, P., Gil, F., Ortiz, S., Puente, y J. Flexible virtual and remote laboratory for teaching robotics f.a. Current Developments in Technology-Assisted Education, 2006.
7. S. CARPIN, M M. Lewis, J. Wang, S. Balarkirsky, y C. Scrapper. Usarsim:a robot simulator for research and education. Int. Conf. on Robotics and Automation, In Proceedings of the IEEE 2007. pp. 1400-1405.
8. JENNY CARTER, Stephen Matthews, y Simon Coupland. Teaching robotics at the postgraduate level: Assessment and feedback for on site and distance learning. Int. Conf. on Robotics in Education, 2012.
9. G. ECHEVERRIA, N. Lassabe, A. Degroote, y S. Lemaignan. Modular openrobots simulation engine: Morse. In Proceedings of the IEEE. ICRA, 2011.
10. S. GALVAN, D. Botturi, A. Castellani, y P. Fiorini. Innovative robotics teaching using lego sets. IEEE Int. Conf. on Robotics and Automation Orlando and Florida, 2006.
11. BRIAN P. GERKEY, Richard T. Vaughan, y A. Howard. The player stage project: tools for multi-robot and distributed sensor systems. Proceedings of the 11th Int. Conf. on Advanced Robotics, 2003. pp. 317-323.
12. M. GINI. Learning computer science through robotics. ASEE Annu. Conf. and Washington, 1996.
13. J. JACKSON. Microsoft robotics studio: a technical introduction. IEEE Robotics & Automation Magazine, 2007. pp. 82-87.

14. JONES, J. L., Flynn, A. M., Seiger, y B. A. Mobile Robots: Inspiration to Implementation (2nd Edition). 1998.

15. J. KRAMER, M. Scheutz. Development environments for autonomous mobile robots: A survey. Autonomous Robots, 2007.

16. E. KROTKOV. Robotics laboratory exercises. Trans. Educ., Feb. 1996. pp. 94–97. IEEE

17. Z. LIANG. Teaching robot vision in manufacturing technology. ASEE Annu. Conf. and Washington and DC, 1996.

18. El libro blanco de la Robótica en España. CEA- GTRob, 2011.

19. EMANUELE MENEGATTI y Michele Moro. Educational robotics from high-school to master of science. Proceedings of SIMPAR 2010 Workshop, 2010.

20. TASKIN PADIR, Gregory S. Fischer, Sonia Chernova, y Michael A. Gennert. A unified and integrated approach to teaching a twocourse sequence in robotics engineering. Journal of Robotics and Mechatronics Vol.23 No.5, 2011.

21. MORGAN QUIGLEY, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, y Andrew Y. Ng. Ros: an open-source robot operating system. In ICRA Workshop on Open Source Software, 2009.

22. VAUGHAN, Richard T., Gerkey, y Brian P. Reusable robot software and the player/stage project. In D. Brugali, editor, Software Engineering for Experimental Robotics, pages 267–289. Springer, Berlin Heidelberg, 2007.