Training master students to program both virtual and real autonomous robots in a teaching laboratory

Elisa Tosello, Stefano Michieletto, Enrico Pagello
Intelligent Autonomous Systems Lab (IAS-Lab)
Department of Information Engineering (DEI), University of Padova
Via Gradenigo 6/B, 35131 Padova, Italy
Email: {toselloe, michieletto, epv}@dei.unipd.it

Abstract—The paper describes how the graduate course “Autonomous Robotics” innovatively introduces robotics to Master of Science students of the Faculty of Computer Engineering of the University of Padova (Italy). The main contributions are: 1) The adoption of a Project-Based Learning constructivist approach. This teaching methodology makes students able to autonomously build their robotic knowledge base; 2) The assignment of laboratory experiences according to an increasing difficulty, from mobile robots (the simple Lego Mindstorms NXT) to humanoids (the Vstone Robovie-X and the Aldebaran NAO). Humanoids are not a widespread teaching tool because of their complexity: the course simplifies the resolution of the robots stability problem by adopting teleoperation; 3) The adoption of the open-source Robot Operating System framework. The framework encourages students to implement reusable code. The effectiveness of the adopted approach has been proven building a team of students that had successfully concluded the course. The team is participating in the European Robotics Challenges and has successfully accomplished the challenge’s first stage.

Keywords—Educational Robotics; Constructivism; Project-Based Learning; ROS; Lego Mindstorms NXT; Aldebaran NAO; Vstone Robovie-X; Humanoid

I. INTRODUCTION

“Autonomous Robotics” (AR) is the title of a Course of the Master of Science (M. Sc.) in Computer Engineering held by the Intelligent Autonomous Systems Laboratory (IAS-Lab) of the Department of Information Engineering (DEI) of the University of Padova (Italy) [1]. The course offers students methodological bases for programming autonomous robotic systems combining theoretical class lectures and practical laboratory experiences. Lectures give students a background on robotic fundamentals; laboratories let them solve robotic problems exploiting the acquired theoretical background. Laboratory experiences are assigned according to an increasing difficulty. First, we ask for the resolution of common robotic problems [2] using a LEGO Mindstorms NXT robot [3] (e.g., path planning with obstacles avoidance and perception). Then, we introduce humanoids and the related stability problem [4]. We provide a Vstone Robovie-X [5] and an Aldebaran NAO [6].

Humanoid robots are a not yet widespread teaching material because of their high cost and complexity. However, the complexity of building and programming humanoid robots make university students in engineering courses interested in studying these platforms [7]. Interfacing with humanoids allows the implementation of intelligent, stable, and balanced multi-Degrees Of Freedom (DOF) movements fixing complex issues like robot stability, multi-limb coordination, and high-DoFs inverse kinematics. Other types of robots, like the wheeled ones, do not offer these features. In our course, we selected two cheap and popular humanoid robots: the Vstone Robovie-X is a 17 DOF robot suitable for first time robotic builders, and the Aldebaran NAO is a platform with 25 DOF, vision, audio, and tactile sensors, usable also by advanced users. Other robotics courses propose the study of humanoids but the adopted hardware is expensive and does not allow an affordable reproduction of their experiments. The University of Tokyo uses the HRP2 [8]; the Shibaura Institute of Technology proposed the E-Nuvo [9], which is less expensive than the HRP2 but still costly. Experiments were conducted to lower costs and make humanoid robots affordable to universities in order to use them as teaching materials. An example is [10], which fabricated robot systems using easily available cheap key components; a servomotor of a toy and a PIC microcomputer, for example. The system is cheap but lacking because it is composed of only robot’s lower limbs. This is useful for a beginner to fabricate his/her first control program, but the study of the robot’s stability is oversimplified. In our approach the stable locomotion problem is not hardware simplified. The simplification is software: we exploit the potential of teleoperation [11]. Compared with existing humanoids teaching assignments [12], teleoperation lets students compare their movements with the robot ones and take advantage of similarities to solve the motion planning problem in a natural “human way”. They can first analyze the human motion, understand how to translate it with respect to the robot constraints, and finally solve the robot stability problem.

We adopt the Project-Based Learning (PBL) [13] teaching methodology, based on the Papert’s perspective of Constructivism [14]. In the lab, students organize themselves in teams and face problems alone freely choosing the resolution method. Students can collaborative discuss, reflect, exchange ideas, and combine each other’s techniques to achieve better solutions. In this way, they develop inquiry, investigation, and collaboration skills, in turn, increasing overall comprehension of the issues [15]. It is the opposite of traditional classrooms embracing the cognitive approach from Neisser [16]: students receive knowledge passively and work primarily alone, learning is achieved through repetition, and subjects are strictly
adhered to and are guided by textbooks. Only few short laboratory experiences are assigned, usually consisting of predetermined instructive sequences solving very particular and simplified real cases. Other robotics teachings adopt the constructivist methodology. An example is the TERECop project [17]. The difference is that institutions joining this project base their teaching only on the LEGO Mindstorms kit. Capabilities of Mindstorms robots are limited and students cannot test their programming abilities solving complex problems. Our labs have an increasing difficulty. At first, three experiences are assigned dealing with LEGO robots; then, three experiences dealing with humanoid robots are presented. As consequence, students first solve simple problems [2]; then, they become comfortable with robotics and can deal with complex robotic problems [4].

The course adopts the open-source Robot Operating System (ROS) robotic framework [18], which collects the most popular robotics libraries. The use of ROS is spreading among the world-wide robotics courses and various robotics challenges foresee its usage to solve service or industrial robotic problems using humanoid or mobile robots (e.g., the ROCKIn@Home [19] and @Work [20] challenges, or the RoboCup@Home and @Work [21] challenges). Other challenges, focused only on humanoids, exist (e.g. the HUMABOT [22], RoboCup Soccer [23] and DARPA [24] robotics challenges). They face challenging robotic problems like robot stability, manipulation or grasping. Since the last decade many robotic frameworks have been developed, e.g., URBI [25], ORocos [26], YARP [27], Microsoft Robotics Studio [28] and Piaget [29]; however, no one has become as widespread as ROS. The adoption of ROS pushes students to develop simple algorithms in a structured environment. They learn to organize software into modules, reuse data structure and classes, and exploit class inheritance.

Usually, only deserving students can participate in a challenge. Our course aims to offer even beginning students the chance to use robots, giving them the necessary background to face European or worldwide robotics competitions. As evidence of the good preparation that the course gives to students, we established a team of students that had successfully concluded the course. They have to address the European Robotics Challenges (EuRoC) [30]. Students, followed by tutors (professors and post-docs), passed the first stage of the competition and received a grant from the European Community.

The rest of the paper is organized as follows. Section II presents the course, describing the laboratory experiences and the examination arrangements. Section III highlights the solutions proposed by the most of the students during the 2011/2012, 2012/2013, and 2013/2014 academic years. Provided solutions aim to facilitate experiments reproduction. Section IV details the stages of EuRoC. Section V exposes student marks obtained at the end of the course (according to the academic years) and results obtained in the EuRoC challenge. In Section VI some conclusions and future perspectives are described.

II. THE COURSE

“Autonomous Robotics” lasts 12 weeks. The attendance does not require any prerequisite: any student can attend it. It is composed of three lessons of two hours per week. During lectures, the teacher explains the theory. Every two weeks the teacher presents a laboratory experience, for a total of six labs, that students have to solve.

A. Lectures

Lectures give an understanding of the relationship between perception and action in natural systems; the comparison between algorithmic techniques and planning; the importance of autonomous learning for the autonomous execution of robotic tasks; the analysis of human movements for the modeling of humanoid robots; and principles of robotic locomotion;

B. Laboratory experiences

Laboratory experiences allow students to learn how to use robotics software and how to solve robotic problems applying algorithms learned in class. They have an increasing difficulty. The first three experiences ask students to work with a simple mobile robot, the Lego Mindstorms NXT. Students have to find a collision-free path leading the robot from a start to a goal points separated by an obstacle. Then, a collision-free trajectory has to be extracted from an assigned map populated by movable obstacles. Lastly, an omnidirectional camera is mounted on the robot and features of the acquired omnidirectional images have to be extracted during the robot navigation. These experiences make students confident with robotics and able to deal with the humanoids project: three laboratory experiences designed to solve the humanoids stabilization and motion planning problems. A VStone Robovie-X has to safety lift an object. Since humanoids stability is a complex problem, the course simplifies its resolution by adopting teleoperation: students have to map human limbs into robot joints and teleoperate the robot guaranteeing its stability during the motion. Moreover, a motion planning problem has to be solved using an Aldebran NAO. Tasks have to be solved in both a simulated and a real environment. ROS has to be used, together with the C++ and Python programming languages. Gazebo is used as simulator [31].

Students organize themselves in teams of two/three (classes of about 20 students per year). Larger groups induce confusion and unbalanced workload division within the group itself. One teacher supervises and helps students when in doubt. No more teachers are required to successfully complete tasks. According to the PBL constructivist approach, the instructor does not slavishly guide students on the fulfillment of assigned tasks: during the class, students learn what the goals are; during labs, they try to solve problems freely choosing the resolution method and discussing with each other. Students can every day access the laboratory and use the available robots.
Details of the laboratory experiences follow. A brief description of every task is depicted, together with the theoretical knowledge required to face it, and the robotic and computer science objectives inferred from its successful completion. Specifics can be found on the Lab Website [1].

The Mobile Robots Project:

1) Obstacle Avoidance: Students have to plan the path of a NXT robot in order to avoid two cones. The robot has to go around the first cone, stop before the second one by exploiting the information arising from a sonar sensor, and come back (see Figure 1).

![Figure 1 – Experience 1: Obstacle Avoidance](image)

   a) Theoretical background: No robotics motion planning algorithm is applied: only the robot’s motion promitts command the planning according to the sensor feedback.

   b) Robotics objectives: The aim is to introduce students to the use of robots and their sensors. They learn how to imprint a motion command and how to interpret and correct inaccuracies. In fact, the NXT sonar sensor is noisy and NXT motors are imprecise. Students become familiar with ROS: they start practicing with ROS master (launch files), they explore nodes and topics functionalities (rostopic and rgraph), they start using the visualizer (rviz), they analyze the publisher/subscriber approaches used by ROS nodes to send and receive messages and exchange data.

   c) Computer science objectives: The goal is to examine the concepts of data structure, class, and object-oriented programming. Students can understand the ROS structure and learn how to develop a customized ROS package using the object-oriented approach promoted by the ROS community in order to build independent, reusable entities that can be assembled in order to control robots.

2) Path Planning: The robot has to navigate into a white squares grid of NxM cells from a START to a GOAL cell while avoiding obstacles (see Figure 2). An infrared sensor is mounted on the robot in order to identify the white squares borders. The robot cannot move diagonally, it navigates using rotations and forward translations by one cell. The experience is divided into two scenarios: the first map is static and obstacles positions are known; the second map is dynamic and obstacles can be moved.

   a) Theoretical background: A basic path planning problem is to build a path that connects a START to a GOAL point while avoiding collisions with obstacles. The Wavefront algorithm [32] finds a path using a Breadth-First Search (BFS) on the graph induced by the neighborhood connectivity (adjacency) of cells, starting at GOAL. As BFS traverses the space, each cell is assigned a value which corresponds to the number of moves required for the shortest path from that cell to GOAL. For adaptive mapping, the robot does not always know where all obstacles are located. In this case, it has to scan the environment after each move it makes, it has to update the map with new or removed obstacles, re-run the Wavefront algorithm, and react to the new updated solution.

   b) Robotics objectives: Students learn more about path planning algorithms and collision-free paths built according to the information acquired by sensors. In fact, the robot has to use sensory information in order to recognize the motion from one cell to another, avoid obstacles, and align the robot to the white borders. Since the second scenario is not known a priori, an internal representation of the robot position and checked cells has to be developed. From the ROS point of view: students learn how to exploit communication between different devices (robot and sensor). Implementing a complex algorithm pushes students to split their code into a reasonable set of modules that should communicate among each other.

   c) Computer science objectives: The experience complexity (e.g., dealing with static and dynamic maps) leads students implementing well-structured code.

3) Perception using computer vision: The robot has to reach a goal on a gridmap with moving obstacles as in Path Planning, but it observes the environment through an omnidirectional camera. Students are asked to find the optimal placing of the camera on the robot and calibrate it. After implementing the vision-based module, able to provide the same output of the infrared sensor used in Path Planning, they have to develop a ROS module capable of analyzing all white stripes visible on the carpet. Closer white lines should be detected in order to evaluate their distance from the robot and to recover the position and orientation of the robot with respect to the grid (see Figure 3).
a) **Theoretical background:** OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library [33]. Its usage allows to graphically process the images acquired by the camera and extract the white lines. Before processing the images, the camera must be calibrated. Calibration allows the computation of a function \( f: \mathbb{R}^3 \rightarrow \mathbb{R}^2 \). This means that given a 3D point \( P(x, y, z) \) coming from the projection center of a mirror, calibration determines how the optical sensor transforms \( P \) into a 2D point \( p(u, v) \). Suppose that the camera and the mirror axes are aligned, then \((x, y)\) will be proportional to \((u, v)\), with \( f \) the function that maps \( p \) in \( P \):

\[
P = \begin{bmatrix} u \\ v \end{bmatrix} = f(u, v)
\]

The mirror is symmetrical, that means \( f \) depends only on the distance between the point and the center of the image:

\[
f(u, v) = f(\rho)\cos \theta = \sqrt{u^2 + v^2}
\]

The best result is reached when the function is of fourth degree:

\[
f = a_0 + a_1 \rho + a_2 \rho^2 + a_3 \rho^3 + a_4 \rho^4
\]

Once \( f \) is found, a deviation from the initial hypothesis could exist because of errors occurred during the setting of the camera and the mirror. The consequence is an ovalization of the outer circular edge of the mirror. In order to keep track of this distortion, the model makes the following affine transformation:

\[
\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} c & d \\ e & f \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} x_c' \\ y_c' \end{bmatrix}
\]

The transformation connects the real distorted coordinates \((u', v')\) and the ideal non-distorted ones \((u, v)\). Calibration computes both the coefficients of the polynomial and the affine parameters. Ocam Calib is a Matlab calibration tool developed by D. Scaramuzza in order to extract the parameters needed to calibrate a vision sensor [34].

b) **Robotics objectives:** Students learn how to deal with vision sensors and vision algorithms, especially with omnidirectional sensors. They learn how to calibrate an omnidirectional camera and how to tilt it in order to extend its detection range. Extending the detection range leads to a more complex processing. Facing the tradeoff between complexity and accuracy is an important expect of this stage. With respect to ROS, this experience combines the image acquisition and processing topics together with localizing and mapping topics inherited from the previous experience.

c) **Computer science objectives:** One of main goals is the creation of a sensor module replacing and improving an existing one. This implies code modularity and encourages the use of inheritance and design patterns. If students choose a good software design in the previous experience, they will be able to easily reuse it in this one.

**The Humanoids project:**

4) **Motion Remapping:** Students have to map human joints into that of a VStone Robovie-X robot. They have to record the human motion using a RGB-D sensor (we select a Microsoft Kinect) and track the human skeleton using a skeletal tracking system, namely NiTE. Skeleton frames must be mapped into robot joints and published over the Transformations and Frames (TF) ROS package to control the robot. The ROS TF package lets the user keep track of multiple coordinate frames over time (e.g., the hip, knee, and ankle reference systems). Simultaneously controlling these systems lets the generation of a robot motion as similar as possible to the human one (see Figure 4).

a) **Theoretical background:** Robot Learning from Demonstration (RLfD) [11] lets a system learn a task performed by a human demonstrator and reproduce it through a robot. A RLfD framework is available to students and can be taken as example [35]. It uses an RGB-D sensor to acquire the scene (human in action). A skeleton tracking algorithm extracts the useful information from the acquired images (skeleton joints poses); and this information is given as input to the motion re-targeting system that remaps the skeleton joints into that of a manipulator robot. After the remapping, a model for the robot motion controller is retrieved by applying a Gaussian Mixture Model (GMM) and a Gaussian Mixture Regression (GMR) on the collected data.
b) **Robotics objectives:** The experience involves motion control, online data elaboration and reaction, human-robot interaction, and teleoperation. Students have to analyze human movements and transpose them into the robot Degrees Of Freedom (DOFs) while dealing with the differences between these motion systems and obtaining a good human motion approximation. They learn how to use the TF ROS module and how to change the reference system while maintaining the fundamental rototranslation constraints. At this preliminary stage, students do not have to consider the robot stability: the robot is fixed to a bracket letting robot limbs move without stability limitations.

c) **Computer science objectives:** Students learn to handle large amounts of data: RGB-D sensors provide RGB and depth images at high frame-rate (30 fps), and the skeleton tracking system provides the joint values recorded at every instant. Students should be able to elaborate the raw data while maintaining an elevate frame-rate in the robot control process.

5) **Robot Stabilization:** The goal is to make a Vstone Robovie-X robot picking up an object by means of human teleoperation (see Figure 5). Using the system developed in Motion Remapping, students have to record the human movements necessary to pick up the object and use them to command the robot. Robot's tilt and fall must be avoided: the theory learned in class about the robot stability control must be applied in order to balance input data. Actions must be performed in real-time: the human has to real-time teleoperate the robot.

   a) **Theoretical background:** The robot's Zero-Moment-Point (ZMP) [36] is the most important factor in implementing stable bipedal robot motions. If the ZMP is located inside the support region, then the robot will not fall down during motions. Moreover, to ensure a stable walk, the robot's Center of Mass (CoM) should maintain the same height during the locomotion [37]. Assume that the motion of CoM is constrained on the surface \( z = c_z \), that \( c = [c_x; c_y; c_z]^T \) is the position of CoM, and that the ZMP is described by the position on the ground \( p = [p_x; p_y; 0]^T \). Then:

\[
\begin{align*}
    p_x &= c_x - \frac{c_x \ddot{c}_x}{g} = c_x - \frac{\ddot{c}_x z}{g} \\
    p_y &= c_y - \frac{c_y \ddot{c}_y}{g} = c_y - \frac{\ddot{c}_y z}{g}
\end{align*}
\]

where \( g \) is the acceleration of gravity. During the execution of the teleoperated movements, the ZMP must be inside the supporting sole (\( p_z = 0 \)) and the CoM has to perform a trajectory perpendicular to the ground. This means \( x = x_c \) and \( y = y_c \). The ZMP position follows.

   b) **Robotics objectives:** Students learn how to teleoperate a robot and how to elaborate a feedback signal able to correct joint values in order to stabilize the robot.

   c) **Computer science objectives:** A collection of concepts learned during previous experiences have to be applied on this environment.

6) **Motion Planning:** Students have to plan the motion of an Aldebaran NAO robot in a 2D simulated environment populated by obstacles. The robot has to walk through the path avoiding collisions with obstacles (see Figure 6).

   a) **Theoretical background:** A motion planning problem aims to produce a continuous sequence of collision-free robot configurations connecting a start configuration START to a goal configuration GOAL. The Open Motion Planning Library (OMPL) [38] is the most commonly used collection of motion planning algorithms. It implements the basic primitives of sampling-based motion planning which, instead of computing the exact solution of the problem, sample the states space of the robot. Examples of available OMPL planners are Probabilistic Roadmap Method (PRM) and Rapidly-exploring Random Trees (RRT).

   b) **Robotics objectives:** Students learn to construct a 2D map and to find a free path through an OMPL algorithm. The experience gained during the Robot Stabilization assignment should be applied to guarantee the robot's stability during the motion.

   c) **Computer science objectives:** This experience will consolidate computer science objectives learned during the course.

D. Examination Arrangements

The final exam of “Autonomous Robotics” consists of two sub-parts:

1) A report of each laboratory experience;
2) A final theoretical and/or experimental project examining a robotics topic.

The final grade goes from 0 to 30 points (max). The 30-point scale is simply divided in two: non passing (0 to 17), and passing grades (18 to 30 cum laude). The laude is assigned to those students who achieve particularly brilliant results. The final grade is computed as follows. The final projects accounts for 80% of the final grade, and each of the two laboratory projects contribute 10% of the final mark. Every experience (three for each laboratory project) contributes 1/3 of the relative project's grade and is mandatory.

Teachers use a 30-point scale to evaluate the final project. Students are asked to make a report and 20 minutes of oral presentation describing their works (15 minutes for presentation and 5 minutes for questions). The expected result is similar to a scientific paper. The ability of defining the
project is necessary to accomplish assigned tasks. No specific algorithm is requested; the implementation has to be loaded in the Lab's YouTube channel. Demonstrating the proper functioning of the designed robotic framework, its novelty, and consequences to which its adoption can lead with respect to service or industrial robotics progresses.

Laboratory projects are evaluated according to the US grading: E (not submitted), D (failed), C (mandatory), B (good), and A (very good). Every report has to include a description of the solution proposed and its source-code. Starting from the 2013/2014 Academic Year, a video demonstrating the proper functioning of the designed implementation has to be loaded in the Lab's YouTube channel. No specific algorithm is requested to accomplish assigned tasks; students can implement a method taught during the lectures or look for a different state of the art technique; they can adopt an existing library or develop a novel algorithm. Reports have to be delivered within three months from the end of the course. The grading considers the technical writing (e.g., document organization, comprehensiveness, style, references, and synthesis), and the complexity and originality of the approach used (e.g., they are rewarded when organizing software into modules, reusing data structures and classes, exploiting class inheritance). Any engineer should possess all these skills. Table 1 and 2 define the grading of each experience.

### III. PROPOSED SOLUTIONS

This section describes the solutions that most of the students proposed during the 2011/2012, 2012/2013, and 2013/2014 academic years.

#### The Mobile Robots Project:

1. **Obstacle Avoidance:** Commands were imprinted to the robot following **Algorithm 1**, where and are distances chosen by the students to avoid cones (see Figure 7).

2. **Path Planning:** The experience is composed of three sub-parts: calculation of the path, motion planning of the robot for moving cell to cell, realignment of the robot.

The Wavefront algorithm was applied to compute the path, both on the static and on the adaptive map (see Figure 8). Without loss of information, students supposed that the robot moved on the map following the polar coordinates (North, South, West, East). Coordinates were obtained from the difference between the coordinates of row and column of the current and next cell. A positive or negative variation of the

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Not submitted</td>
<td>Not submitted</td>
</tr>
<tr>
<td>D</td>
<td>The robot does not move</td>
<td>The robot does not move cell to cell because of the realignment failure</td>
</tr>
<tr>
<td>C</td>
<td>The robot moves</td>
<td>The robot moves but it does not reach the GOAL</td>
</tr>
<tr>
<td>B</td>
<td>The robot imprecisely solves the task (e.g., motors are not calibrated, movements are noisy)</td>
<td>Path found only for the static map</td>
</tr>
<tr>
<td>A</td>
<td>The robot precisely solves the task</td>
<td>Path found both for the static and for the adaptive map</td>
</tr>
<tr>
<td>A+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 1 – Marks of the Mobile Robots Project**

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Not submitted</td>
<td>Not submitted</td>
</tr>
<tr>
<td>D</td>
<td>The robot does not move (e.g., the tracking system does not track human joints, human and robot joints do not match, the ROS publisher node does not publish the TF values necessary to move the robot)</td>
<td>The robot falls (e.g., no stabilization rule is applied or the stabilization feedback signal is incorrectly implemented)</td>
</tr>
<tr>
<td>C</td>
<td>The robot moves</td>
<td>The robot performs the movement but swings</td>
</tr>
<tr>
<td>B</td>
<td>The robot moves and joints match</td>
<td>Only unjustified manual corrections aim to stabilize the robot</td>
</tr>
<tr>
<td>A</td>
<td>The robot reproduces the human</td>
<td>The system is stable</td>
</tr>
<tr>
<td>A+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 2 – Marks of the Humanoids Project**
line index corresponds to a shift towards the South or the North; an increase or decrease of the column index corresponds to a shift towards the East or the West. An array contained the list of coordinates necessary to reach the GOAL following the computed path. To move from one cell to another, the robot followed commands developed in Obstacle Avoidance: it went forward until finding the white line, and it rotated by a certain angle \( \theta \) according to whether the robot should be oriented toward the North, South, West, or East with respect to the current position.

Because of imprecise movements, students checked the alignment of the robot with respect to the edges of the cell at each transition to a new cell (see Figure 9). In detail, the robot advanced in order to cross the line and make the infrared sensor able to detect the tilt angle of the robot. A left rotation by a certain angle \( \alpha \) (until the line) impressed powers to the motors. The robot returned to the starting position. A right rotation by a certain angle \( \beta \) moved the robot until the other side of the line. In this way the robot made a rotation of 180°, equal to the sum of \( \alpha \) plus \( \beta \); the arithmetic mean of these two values returned the angle to be performed by the robot in order to be oriented perpendicular to the line.

3) Perception Using Computer Vision: Students used the omnidirectional camera in order to take a picture of the robot’s surroundings. From this picture they extracted 2D points describing the lines of the grid. These points were remapped into the 3D space in order to reconstruct the real world around the camera. The experience was divided into fourth sub-parts: the camera calibration, the image processing, the extraction of the points mapping the grid on the image, the conversion of these points into the 2D points of the real world. Calibration was done using OcamCalib as follows: students used the camera in order to acquire a set of pictures of a chessboard. Pictures sequentially covered the entire surface around the mirror (see Figure 10.a). They computed the angles of the squares of the chessboard in order to obtain an oriented Cartesian reference system (see Figure 10.b). Coefficients of the fourth degrees function \( f \) were computed and the tool was calibrated. After the calibration, the camera took a picture of the environment. In order to better distinguish white lines, the RGB image with three channels was turned into a binary image with a channel (black-and-white) and the remaining impurities were removed using OpenCV functions (see Figure 11.b). Given the black-and-white image free of impurities, it was elaborated in order to find the grid. In detail, a range finder was simulated in order to totally scan the image: every radius started from the center of the camera, had a distance \( \theta \) from the previous one, and stopped when it encountered a white line (see Figure 11.c). For every encountered white point, the algorithm checked its non-isolation to avoid the storage of impurities. The coordinates of every non-isolated point were stored on a queue. Once given the coordinates of the white points composing the grid lines and given the function \( f \), students transposed the 3D coordinates of the camera into a 2D plan and reproduced the grid.

**Algorithm 1: Simple Obstacle Avoidance**

1. Move forward and stop \( x \) cm before the cone;
2. Rotate 90° to the right;
3. Go \( y \) cm forward;
4. Rotate 90° to the left;
5. Go \( z \) cm forward;
6. Rotate 90° to the right;
7. Go \( y \) cm forward;
8. Rotate 90° to the left;
9. Go forward and stop \( x \) cm before the cone;
10. Rotate 180°;
11. Repeat from 1.
The Humanoids project:

4) Motion Remapping: The mapping experience was mainly faced by using two methods. The first one matched each robot joint with that of the human counterpart and computed the angle of every joint. The resulting values were used to properly move the robot. The matching phase could be very tricky because the robot kinematic chain could be very different from the human one, especially in some angle limits. Some students looked for the maximum and minimum of each selected human joint by testing several subjects; then, they scaled the computed joint values according to the limitation imposed by the robotic platform. Other groups had to face a singularity problem in the selected mapping and proposed an hysteresis system to prevent rapid switching of configuration in the humanoid due to sensor noise. This method usually results very natural to users, on the other hand it is not so precise because a small error in an angle could correspond to a huge change in position. A second method was used to avoid this behavior: inverse kinematics. Students identified a sort of end effector for each limb and computed the joint angles in order to obtain similar positions between human and robot. Nevertheless, the similarity is limited to the end effector position while other joints can assume very different configurations with respect to the human body, so in some cases the robot motion seems quite unnatural to users. During the academic year 2011/2012, a team tried to overcome limitations of the already presented methods by mixing them in a hybrid solution. Like in the first method, to ensure the motion to be as natural as possible, they fixed some joint angles by matching human and robot joints. While the remaining ones were computed by means of inverse kinematics in order to reach a good precision of the end effector position. The result has proven to be quite effective and it has been widely adopted from almost all the students of the following years.

5) Robot Stabilization: Students kept the CoM projection of the robot inside its support region. A simple solution imposed a customized relation among the hip (α), knee (β), and ankle (γ) joints (i.e. γ = α = β/2) in order to balance the robot. More complex solutions involved all the lower body joints or the entire structure of the robot. These solutions made the robot motion more natural. Dynamic methods involved the gyroscope to measure the robot inclination and balance it by applying an appropriate motion: the platform was modeled as an inverted pendulum and the stability was guaranteed by compensating forces causing the robot fall with forces able to move the torso in the opposite direction. This technique is particularly suitable for the picking up action, in which the object mass has to be considered part of the robot in order to maintain its stability.

6) Motion Planning: Students built their own map and made the robot navigate from a point to another within it. Most students used the RRT algorithm of the OMPL library. Some groups built a 3D map representing a real environment, some other tested the navigation using the real NAO platform.

IV. EuRoC

EuRoC is a 4-years long program that aims to develop competitive solutions to improve the European manufacturing industry. In order to make European manufacturing enterprises, in particular Small Medium Enterprises (SMEs), able to adapt to global competitive pressures, they should adopt robotics-manufacturing systems able to efficiently and sustainably adapt to the changing industrial environment. In this context, EuRoC proposes the following challenges.

1) Reconfigurable Interactive Manufacturing Cell: The challenge involves manufacturing supply chains within the production stage. The goal is to develop novel manufacturing applications allowing the robust assembly in presence of part tolerances and flexible parts. The challenge includes the development of adaptive perception and cognition skills needed for guarantying the good functioning of the system in case of complex work cell layouts and dynamic environments, illumination changes and tolerances. The challenge requires the human-robot collaboration and the implementation of distribution algorithms that should dynamically and real-time reconfigure the assignment of tasks among human workers and robot arms in order to account for bottlenecks, breakdowns, etc.

2) Shop floor logistics and manipulation: The focus is logistics. Robots have to autonomously navigate and operate in unstructured environments.

3) Plant servicing and inspection: Servicing is the challenge. The goal is to develop solutions that enable Micro Air Vehicles (MAVs) to be used in real life scenarios.

A team composed of research experts (academia), solution-oriented or innovating companies (industry) can apply to at least one of these challenges and it has to pass the following three stages to win.

1) Stage I – Qualifying:

a) Part A – Call for challengers: Every team has to submit an outline including the team description and the challenge to which it intends to participate. According to the selected challenge, teams have to solve a simulation test set by the host. A score is assigned to every task of the test according to objective metrics (e.g., the success of the assignment). The best 15 teams of every challenge (3x15) are selected.

b) Part B – Advance to Stage II: Challengers team up with end users, technology developers, and system integrators propose a customized robotic solution able to improve the end-user production, logistics or servicing processes. Challenge Advisors and independent experts evaluate the novelty of the proposed solution. The best 3x5 teams are selected.

2) Stage II – Realistic Labs:

a) Part A – Benchmarking and freestyle: Teams have to overcome a benchmarking stage in which the host assigns a task to be performed by a real robot. Simultaneously, they have to overcome the first part of their proposal, namely freestyle. The Challenge Advisory Board and a team of independent experts decide admission to Stage II Part B.
b) Part B – Showcase: Teams implement the overall proposed solution and test it under realistic conditions. Advisory Board and independent experts will evaluate and rank Challenger teams. The best 3x2 teams are selected.

3) Stage III – Field tests: Teams perform experiments under real conditions at end-user sites. After a final evaluation by a Board of Judges, the EuRoC winner is selected.

Every challenge requires the use of ROS as the underlying middleware, conforming the usefulness of the framework in the industrial field.

V. RESULTS

The graph of Figure 12 reports the percentage of students having final grades from 18 to 30 cum laude during the three academic years considered in this work (Figure 12.a, 12.b, and 12.c). The good trend persists. Considering all students attending the course in the three academic years (Figure 12.d), more than 30% of them obtained the maximum mark (30L). No student got a fail. The lowest mark was 24, assigned to 10% of the students.

Achieved results prove that adopting a Constructivist teaching approach make students able to apply the knowledge gained in class to solve real problems; they became able to autonomously select the better resolution approach, and to structure data in order to efficiently apply coded functions and methods in different situations.

The effectiveness of the adopted approach has been proven involving about 10 students in the EuRoC Challenge 1. Students were collected among those who successfully concluded the “Autonomous Robotics” course of the 2013/2014 Academic Year, and were team up with IT+Robotics, a University Spin-off. The team is still involved in the challenge. It has been teamed up also with a system integrator and an end-user, and has received a grant from the European Community by proposing to develop a learning-based approach. These latest good results confirm that the adopted teaching approach is effective, and prove that adopting ROS as robotics teaching framework make students ready for the robotics job market.

The best way to show the effectiveness of the adopted approach is still making a survey among students. At the end of the course, students were asked to fill an anonymous questionnaire.

Table 3 collects the most significant questions. Students could choose among four states: Not at all (yellow), A little (red), Enough (blue) and Very much (green).

The questionnaire tested:
- skills acquired in terms of programming capabilities and team working;
- the ability to solve real robotic problems;
- the closeness with future jobs.

Answers to the questionnaire highlight similar results for all the considered academic years. Students programming capabilities improved (Q6). ROS forced students to well structure their software and make it reusable (Q5). Students understood the potential of team working in achieving complex results (Q2, Q3). They showed a moderate confidence on the fact that expertise coming from lab experiences could be reused in a future job (Q7), but wished more courses adopting this approach (Q4, Q9). Acting in this way, a new class of young, versatile engineering is ready to enter the job market (Q8).

VI. CONCLUSIONS

The paper presents the “Autonomous Robotics” course of the M.Sc. in Computer Engineering held at the Scholl of
Engineering of the University of Padova. The paper focuses on a set of laboratory experiences proposed to students adopting the Constructivist teaching approach, and exposes how students that had successfully concluded the course got involved in the European Robotics Challenges.

The correct resolution of the assigned tasks and the positive results obtain from EuRoC give instructors the certainty that combining Constructivism with a gradual increase of the level of difficulty of assigned tasks is effective in teaching robotics. Moreover, obtained results confirm that teaching ROS prepares students for the robotics engineering profession.

Our goal for the future is to expand the teaching framework including new sensors and robot functionalities. The expansion will make the course a solid foundation that will train students for the robotics working world.

REFERENCES

[1] Intelligent Autonomous Systems Lab [online], http://robotics.dei.unipd.it/


