Experiments with a Virtual Lab for Industrial Robots Programming

http://dx.doi.org/10.3991/ijoe.v11i5.4752

P. Abreu, M. R. Barbosa and A. M. Lopes
University of Porto, Porto, Portugal

Abstract—**This paper presents the use of a virtual lab for teaching industrial robots programming to university students. The virtual lab, that replicates the existing physical lab, is built using an industrial simulation software package, RobotStudio™. The capabilities of this tool are explored in order to complement the introduction of theoretical concepts with practical programming experience. In addition to illustrate the use of different coordinate systems in a robotic cell, a description of the tool center point calibration and examples of evaluating different moving strategies to cover a plane surface, are also presented.**

Index Terms—**Robot programming, Robotics, Student experiments, Virtual labs.**

I. INTRODUCTION

Industrial robotics is a key automation technology used extensively and increasingly in production facilities [1]. Among the multiple adopters of these systems, the automotive industry is a major user both in terms of number of robots installed and diversity of applications. Examples of these industrial applications include multiple processes such as manipulation, palletizing, welding, painting, deburring and others [2]. To program the robots to perform these tasks a fundamental requirement, when setting up a robotic cell, is to minimize the time the robots must be taken out of production. Particularly in flow line production, as it is often the case of automotive production lines, the downtime represents high costs. The adopted strategies to deal with these problems include using off-line programming software tools. These simulation tools use virtual models of the robots and other elements of a working cell to provide a programming environment for creating, testing and validating the programs for the real cell.

The use of off-line programming to create virtual labs is particularly useful when we consider the requirements for teaching robot programming at an undergraduate engineering course. They represent the possibility of overcoming the restrictions, imposed by the limited number of industrial robots normally available, and allow for multiple students to practice at their own time and pace. In addition, safety issues concerning both the students and equipment are naturally a major advantage of virtual environments comparatively with using real robotic cells.

Virtual labs have been widely used in education and training [3-5], namely in the field of robotics and automatic control [6-13]. These tools are frequently adopted as a complement to the classical teaching/learning approaches [14].

There are various general purpose and proprietary off-line programming software packages for industrial robots [15]. It is possible, using these systems for the user, i.e. student, to build and configure a virtual robotic cell, program the robot and simulate the robotic process. The program can then be taken to the physical robotic cell for implementation. Typically, at this stage, it is necessary to conduct calibration routines which naturally should be accomplished in a shortest time as possible. Off-line programming is a powerful approach to robot programming. However the learning curve to allow an effective use of the software may pose some restrictions, particularly in cases of academic robotic courses where the time available must be adequately allocated, both for an analysis of theoretical concepts and practice sensibility. To overcome this limitation our approach uses a set of predefined robotic cells created using a specific industrial robot programming software, RobotStudio™ from ABB [16]. This strategy enables the students to work with industrial software, increasing their motivation, as well as using a virtual model of the real robotic cell available in the course.

The paper is organized as follows. Section II, describes the robotics course objectives and the resources available. In Section III, the software used and the implementation of the robotic virtual cell are introduced. In section IV the different learning activities are illustrated with three particular examples of virtual experiments. Finally, Section V presents the conclusions and discusses further developments.

II. ROBOTICS COURSE OBJECTIVES AND FACILITIES

The development of virtual labs for teaching the subject of industrial robotics is intended for the Industrial Management (MIEIG) degree at the Faculty of Engineering of the University of Porto, Portugal. In this degree the industrial robotics subject is taught within the class on Industrial Automation Systems (5 ECTS) at the second year. Robotics is only one of the various automation subjects of a more general course. In modern manufacturing systems robotics solutions are natural candidates for an automation alternative as they provide the potential for increasing flexibility levels, even more with the evolution trend towards using robots capable of sharing the same environment with a human operator. However the current level of development of industrial robotics still requires a clear understanding to fully appreciate the flexibility objectives attainable in an industrial environment. Therefore management and industrial engineers, which normally are not concerned, or do not have the time, to understand the details of robotic technology, must be provided with effective experience and knowledge of the implications of using industrial robots in an industrial application.

To cope with a high number of students (around eighty) the course is organized to include theoretical lectures and lab sessions. The first are devoted to introduce concepts and principles on robot applications and their programming. In the current lab sessions the students, organized in groups not exceeding twenty elements, are able to observe a demonstration of the use and programming of an industrial robot. The robotic cell is built around an ABB IRB2400 robot with an IRC5 controller. It includes an automatic tool change mechanism, three pneumatic grippers, an autonomous spindle and a one axis turning table. The cell is fitted with physical barriers and a light curtain for safety purposes. The labs demonstrations are complemented with sessions were the students use a virtual robotic cell based on the existing real lab. Using this virtual cell, described in the next section, students are able to practice with the details of operating and programing an industrial robot. This allows students to work at their own pace and therefore complement the experience gained on the taught lab sessions.

III. ROBOTIC VIRTUAL CELL

The robotic virtual cell, that reproduces the existing real cell, is part of a virtual lab that includes other cells developed with the software RobotStudio™ [17]. This virtual cell allows replication of procedures used in the real cell such as robot jogging with the FlexPendant, which will be used for the calibration of the tool center point. Simulation and evaluation of different path strategies is also explored with the virtual cell.

The elements modelled in the virtual cell (Fig. 1) are the ABB IRB2400 robot with an IRC5 controller, the rotating table IRBP500C, the pneumatic tool change mechanism and three different grippers from Schunk™. These grippers are modelled as a mechanism to enable simulation of the fingers opening and closing movements. The robot controller is configured to allow the operation of the gripping mechanism through digital output signals, in a way which exactly replicates the configuration on the real controller. In order to replicate this model, of the real controller and all the elements of the cell, in any computer running RobotStudio™, the software provides a specific saving functionality - *Pack and Go*. This is particularly useful to provide students with the same simulation environment, independently of being in the lab sessions or using their own computer.

RobotStudio™, which has been used since 1998, is an off-line robot programming and simulation tool developed by ABB to support their industrial robots. It enables the graphical programming of robots, editing, debugging and simulation of programs. Other features include verifying robot accessibility, reach, collisions and analysis of cycle time, in a robotic cell. The software includes a library of mechanisms and components from ABB, with the possibility of being extended to include other user defined components.

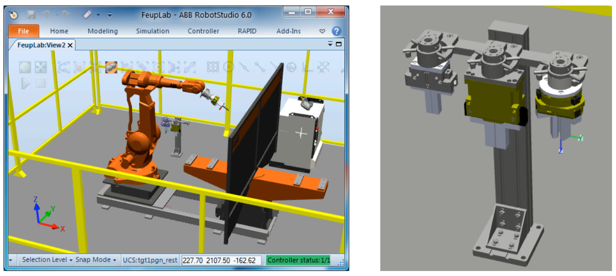


Figure 1. Virtual robotic cell and tools.

One key aspect of this software, for obtaining an accurate off-line programming environment, is the use of what ABB calls “virtual robot technology”. This technology implements a *Virtual Controller* on the PC that runs the same code of the physical controller of the robot. The program developed in this off-line environment can then be directly downloaded to the real controller, without any need for post-processing. The model of the robot controller includes both kinematic and dynamic behaviors, enabling for an effective use of the simulation.

Other features of the RobotStudio™ include the use of CAD based programming functions (i.e. *AutoPath*) that based on the geometric model of parts, generates automatically the robot positions to be used by the robot program. This feature is particularly suitable for programming robot paths on geometrically complex parts. The user, after creating the part or importing it from other proprietary CAD programs such as Solidworks™, just has to select the geometric features on the part that enable the automatic generation of a series of path positions required to accomplish the path with a given tool orientation. The user can examine the tool position at each created target and modify the orientation if necessary.

Another useful functionality provided by RobotStudio™ is record and analyze different controller parameters such as the position, velocity and acceleration, expressed both in terms of individual joints or Cartesian workspace. Other signals, recorded from the *Virtual Controller* information stream, include digital inputs, outputs and the total energy consumption. These monitoring facilities are particular interesting for evaluation and comparison of different strategies to accomplish the same tool path.

RobotStudio™, being developed to support industrial users of ABB robots, can be considered as a professional tool. It can also be used at the education level since ABB offers special conditions for students and universities. Furthermore, a fully functional version has been available for download, free of charge and running for an experimental period of thirty days (http://developercenter.robotstudio.com).

RobotStudio™ is recognized as a valuable industrial automation tool and a motivating factor for students to engage in the learning process. The simulated applications allow the testing of close to real robotic cells, while avoiding the risks, both for the students and for the robots, of operating on real physical systems.

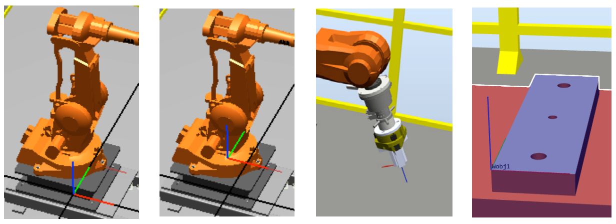
IV. LAB ASSIGNMENTS AND EXPERIMENTS

In this section the three selected examples of lab assignments are presented. The first one focuses on the use of robot jogging on different coordinate systems, replicating on the virtual robotic cell the procedures that are used on on-line programming. The second one describes the calibration procedure for tool center point definition and introduces the associated mathematical model. The last assignment evaluates the robot path trajectory for different moving strategies to cover a plane surface placed in different locations within the workspace, taking into consideration the time, the velocity profile, and the motors energy.

The first example is a lab assignment performed at a lab hands-on session. This initial example illustrates the elementary function to move (jogging) the robot in space using the robot FlexPendant. This basic function is used to introduce the different coordinate systems available to jog the robot and the advantages of selecting the most appropriated one according to a specific desired movement.

The student starts by opening the virtual robotic cell with predefined paths and identifies the defined coordinate systems used: world coordinate system, robot coordinate system, tool coordinate system and the coordinate system associated to an external workpiece (Fig. 2). To jog the robot, the student is instructed to open the Virtual FlexPendant (Fig. 3) which provides access to select a specific coordinate system. Acting on the joystick the student moves the robot accordingly.

The use of the jogging facility let students practice the operation of moving the robot in joint and Cartesian spaces. It is also possible to appreciate the difficulty that occurs when it is necessary to place the robot tool in a desired location, which is required during on-line programming. Understanding the differences between coordinate systems and the procedure to define or edit them can only be fully appreciated through experimental practice in a real cell. The virtual cell provides identical practice if the risks of working with the real systems are ignored.



(world cs) (robot cs) (tool cs) (workpiece cs)

Figure 2. Coordinate systems (cs) used in the virtual cell.

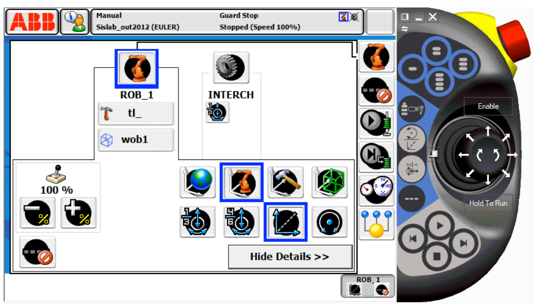

Manual
Sislab_out2012 (EULER)
Guard Stop
Stopped (Speed 100%)

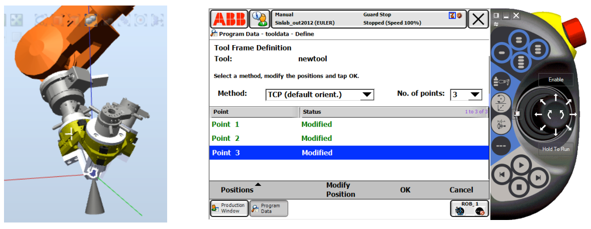
Figure 3. Virtual FlexPendant.

The second example is a lab assignment performed partially at the lab hands-on session and followed by work carried out outside the lab. This assignment presents different methods that can be used to calibrate the tool center point (TCP) of a given tool. The TCP is the origin of the frame associated to the tool. It is important to define correctly this frame because the TCP is used to establish the robot trajectory. The TCP frame is expressed relatively to the tool frame (Tool0) defined on the robot end-effector by the manufacturer. To define the TCP, it is possible the use of different methods. One of them involves specifying the position and orientation of the tool frame, based on the geometry of the tool. This method is particularly suitable to be used with off-line programming, since it is easy to define and identify a frame on the existing geometric model of the tool. Naturally, this method implies that the tool must be correctly modelled. Another method used with on-line programming and provided by robot manufacturers, involves using a pinpoint probe as a calibration target and executing a calibration procedure. This procedure requires placing the probe inside the robot workspace so that the robot tool can reach that point. Normally the robot is manually jogged to position the tool on the pin point using different orientations, at least three and up to nine. A calibration routine from the robot controller is then executed returning the position of the TCP and the respective tool coordinate system.

The students are instructed to use this method on the virtual cell to identify the TCP of one of the available grippers (Fig. 4). To implement this method the model of a cone probe is provided and the robot is jogged to the tip of the cone with three different orientations. The robot positions are recorded using the FlexPendant calibration routine and the results are obtained.

Students are encouraged to explore this procedure, repeating the process using different number of points with different robot orientations and compare the results with the one obtained from the geometrical model. The TCP position, based on the geometrical model is X=0, Y=0, Z=298 mm. Students typically obtain results that are out of the geometric model in a range up to 2 mm. These results are discussed and explained based on the difficulty and uncertainty that exist in placing the tool in the same point with different orientations.

These experiments with the virtual cell on the TCP definition are used to introduce the mathematical concepts required to implement the calibration routines the robot controller provides. The mathematical modeling of the TCP calibration uses robot kinematics coupled with a geometry constraint. The solution of the model involves, generally, an optimization procedure that minimizes a cost function defined in the model. Using the geometry constraint as a point that is reached with different robot orientations, the TCP can be obtained using a least squares


Manual
Sislab_out2012 (EULER)
Guard Stop
Stopped (Speed 100%)
Program Data - tooldata - Define
Tool Frame Definition
Tool: newtool
Select a method, modify the positions and tap OK.
Method: TCP (default orient.)
No. of points: 3

Point	Status
Point 1	Modified
Point 2	Modified
Point 3	Modified

Figure 4. Tool frame definition at the pinpoint using three points.

algorithm. This is the method that is adopted by many robot manufactures that uses the robot as a measurement tool. The general calibration model [18], when considering only the calibration of the tool position and ignoring the orientation, is given by:

$$\min\left\|f(\mathbf{t}_{gi})\right\| \quad \text{s.t.} \quad \mathbf{t}_{gi}=\mathbf{R}_{0i}\mathbf{t}_{vi}+\mathbf{R}_{0i}\mathbf{t}_{s}+\mathbf{t}_{0i} \qquad (1)$$

where,

$f(\mathbf{t}_{gi})=0$ represents a geometry constraint which is a linear or nonlinear function of $\mathbf{t}_{gi}$;

$\mathbf{t}_{gi}$ represents the position vector of the tool tip in the measurement i, relative to the robot base frame;

$\mathbf{R}_{0i}$ represents the orientation matrix of the robot end-effector (Tool0);

$\mathbf{t}_{0i}$ represents the position vector of the robot end-effector (Tool0), obtained with the direct kinematics of the robot;

$\mathbf{t}_{vi}$ represents the position of the calibration target relative to the tool frame;

$\mathbf{t}_{s}$ represents the tool position relative to the robot end-effector (Tool0).

When using a point constraint, $f(\mathbf{t}_{gi})=0$ represents a linear constraint:

$$\mathbf{t}_{gi}=\mathbf{t}_{g(i+1)}=\mathbf{t}_{g} \qquad (2)$$

where $\mathbf{t}_{g}$ is a fixed position within the workspace of the robot. Using (2) into (1) and considering that $\mathbf{t}_{vi}=0$, the following linear equation is obtained:

$$\mathbf{R}_{0i}\mathbf{t}_{s}+\mathbf{t}_{0i}=\mathbf{R}_{0(i+1)}\mathbf{t}_{s}+\mathbf{t}_{0(i+1)} \qquad (3)$$

With (3) the vectors $\mathbf{t}_{g}$ and $\mathbf{t}_{s}$ (the TCP) are obtained. When multiple points are used (3) represents an overdetermined linear equation that can be solved with a linear least squares algorithm. This equation can be written in the form:

$$\begin{cases}\mathbf{Ax}=\mathbf{b}\\(\mathbf{R}_{0i}-\mathbf{R}_{0(i+1)})\mathbf{t}_{s}=\mathbf{t}_{0(i+1)}-\mathbf{t}_{0i}\end{cases} \qquad (4)$$

So that $\mathbf{t}_{s}=(\mathbf{A}^{T}\mathbf{A})^{-1}\mathbf{A}^{T}\mathbf{b}$ with,

$$\mathbf{A}=\begin{bmatrix}\mathbf{R}_{01}-\mathbf{R}_{02}\\ \dots\\ \mathbf{R}_{0i}-\mathbf{R}_{0(i+1)}\end{bmatrix} \text{ and } \mathbf{b}=\begin{bmatrix}\mathbf{t}_{02}-\mathbf{t}_{01}\\ \dots\\ \mathbf{t}_{0(i+1)}-\mathbf{t}_{0i}\end{bmatrix},$$

where $(i+1)$ is the number of measurements made with the robot in different orientations.

The solution that minimizes $\left\|\mathbf{At}_{s}-\mathbf{b}\right\|^{2}$ is:

$$\mathbf{G}(\mathbf{t}_{s})=\sum_{i=1}^{N}\left\|(\mathbf{R}_{0i}-\mathbf{R}_{0(i+1)})\mathbf{t}_{s}+\mathbf{t}_{0(i+1)}-\mathbf{t}_{0i}\right\|^{2} \qquad (5)$$

The obtained value of $\mathbf{t}_{s}$ can be interpreted as representing the radius of a sphere that fits the multiple measured points. The students are then requested, out of the lab sessions, to use these insights to create a program that implements a TCP calibration routine.

The third example looks at the implications on using different robot path trajectories to accomplish a given task in terms of trajectory, time and energy consumption. Three different robot path trajectories are considered to move the robot along a planar rectangular surface.

These trajectory requirements can be associated to different processes such as painting, glue deposition or machining. However, in the case of this assignment, the focus is on the trajectories rather than on the processes specific parameters. Three different TCP trajectories are considered: zigzag, parallel spiral and true spiral (Fig. 5). These trajectories are defined on a planar surface with 200x100 mm so that a tool with 10 mm diameter can cover the surface, considering a step over distance of 7.5 mm. The total mass carried by the robot is 15 kg. The specified linear velocity for all trajectories is set to 100 mm/s. The program of the robot to run each trajectory is already available in the virtual model. In order to compare the different trajectories, the approach and retract movements of the robot TCP are not considered. As such, the starting and ending points of the trajectory lie on the surface. Each trajectory is defined by a given number of targets using linear and circular interpolation moving instructions. In performing a moving instruction, it is possible to define the closeness of the robot in reaching the programmed target by defining the parameter *zone* (Z) available in the robotic language. The paths use the parameter *zone* set to Z1. This means that the robot does not stop in the target but approaches it with a maximum deviation of 1 mm. This setting assures a smooth motion in the vicinity of the target points that define the trajectory.

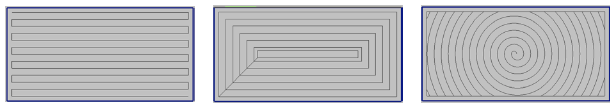

(zigzag) (parallel spiral) (true spiral)

Figure 5. Test path trajectories.

The students are required to run the program for each trajectory and, using the functionalities of the *Signal Analyzer*, record the data relative to time, velocity and total motor energy. Table I presents the compiled data complemented with the number and type of moving instructions as well as the total travel distance.

TABLE I.
DATA FROM THREE DIFFERENT PATH TRAJECTORIES

Trajectory	Zigzag	Parallel spiral	True spiral
Travel Time [s]	26	28	47
Total motor energy (TE) [J]	1515	1598	2806
Max. linear velocity [mm/s]	100	100	100
Number and type of move instructions	26/MoveL	35/MoveL	12/MoveL 54/MoveC
Travel distance (TD) [mm]	2560	2748	4685
Ratio TE/TD [J/mm]	0.592	0.582	0.599

In global terms, the data from this table shows that the travel time is similar when using the trajectories based on using only linear moving instructions (zigzag and parallel spiral). Regarding the strategy that combines the use of linear and circular moving instructions (true spiral) the increase in travel time can be explained by the repetition of the linear movements when reaching the boundaries of the surface. Considering the ratio of total motor energy by the travel distance it is possible to conclude that the different moving strategies present similar values. In addition to these global parameters, the S*ignal Analyzer* provides detailed information on the path trajectory to be used for comparing the different moving strategies. Fig. 6 shows the velocity profile of the three trajectories. It can be seen that the path velocity drops when the trajectory changes direction. As expected, the true spiral strategy presents a smother path velocity until the geometric limits of the part are not reached. When looking in more detail at the velocity profile transitions, it can be observed the limitations on the robot path velocity imposed by the path length. Fig. 7 presents the path velocity during the first turning segment in the zigzag trajectory. As the robot approaches the first turning point, (P) in path graph (2), the velocity drops. The path between (P) and (Q) is not long enough (7.5 mm) to allow the robot to reach the programmed velocity of 100 mm/s. If the same path trajectory is performed at a lower velocity (e.g. 50 mm/s), the robot is able to reach the path velocity at the same turning segment (Fig. 8), although it still occurs a drop in the velocity as the robot approaches the turning points.

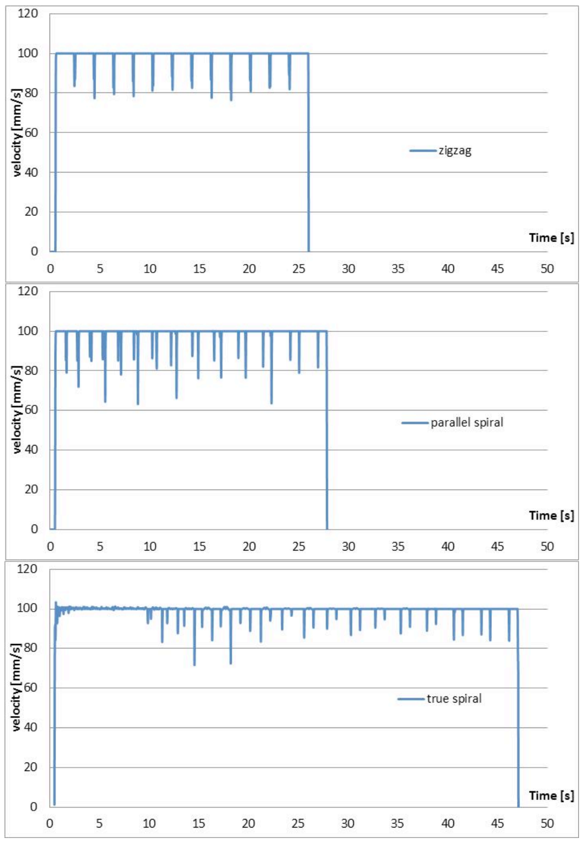
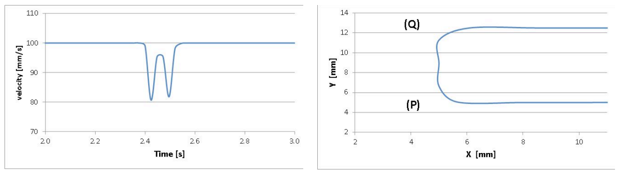
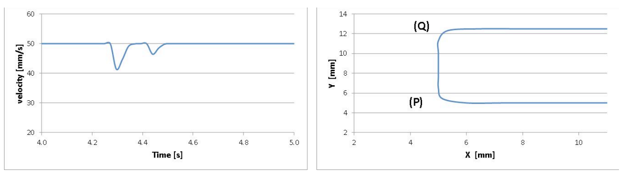


Figure 6. Velocity profile along the three path trajectories: zigzag, parallel spiral and true spiral, respectively.



(1) (2)

Figure 7. Velocity profile (1) and path (2) along the zigzag trajectory during the first turning point performed at a programmed velocity of 100 mm/s.



(1) (2)

Figure 8. Velocity profile (1) and path (2) along the zigzag trajectory during the first turning point performed at a programmed velocity of 50 mm/s.

The use of the *Signal Analyzer* within the simulation program provides the means to explore the robot behavior in these different path strategies. To view the influence on the motor energy used by the robot in a given trajectory, students are instructed to change the position of the path trajectory within the robot workspace. Fig. 9 presents the two locations within the robot workspace where the zigzag path trajectory is implemented: location (a) is in the center of the robot workspace while (b) is closer to the boundaries.

Fig. 10 and Fig. 11 present the graphic from RobotStudio™ *Signal Analyzer* where the data from the measurement of the total motor energy is plotted. As it can be seen from the plots, the energy the robot uses is dependent on the location of the path, being higher for the path performed closer to the working space boundaries, having a value of 4456 J versus 1515 J for the same path performed at the central location. This is due to the fact that when the robot is closer to the boundaries the arm is fully extended and the dynamic load is higher.

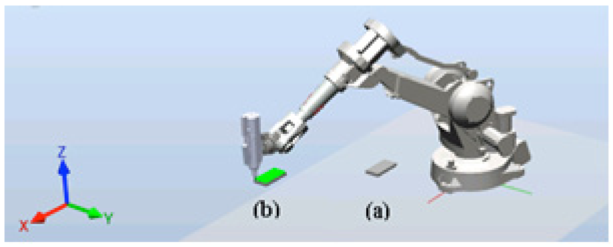
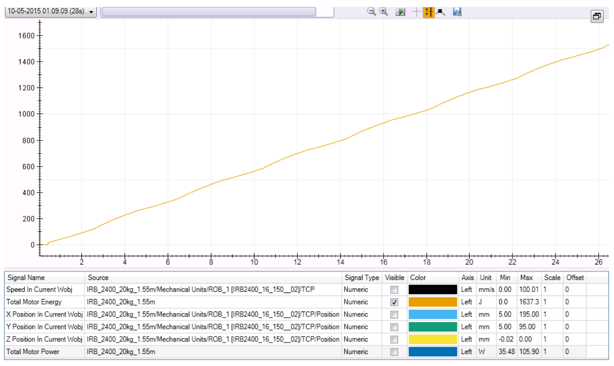
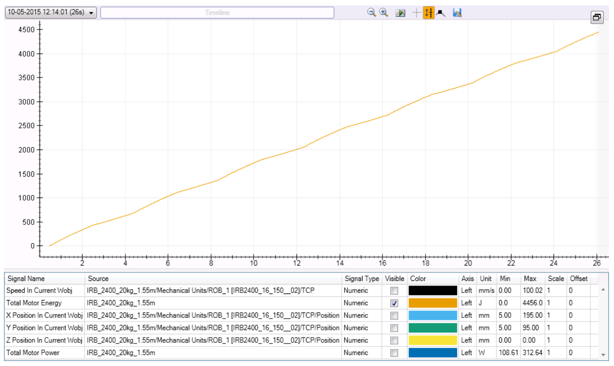


Figure 9. Locations for path implementation: (a) = [830, -100, 420] and (b) = [1530, -600, 420], expressed in mm relative to the robot cs.



Signal Name	Source	Signal Type	Visible	Color	Axis	Unit	Min	Max	Scale	Offset
Speed In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP	Numeric	☐	⬛	Left	mm/s	0.00	100.01	1	0
Total Motor Energy	IRB_2400_20kg_1.55m	Numeric	✓	🟧	Left	J	0.0	1637.3	1	0
X Position In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP/Position	Numeric	☐	🟦	Left	mm	5.00	195.00	1	0
Y Position In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP/Position	Numeric	☐	🟩	Left	mm	5.00	95.00	1	0
Z Position In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP/Position	Numeric	☐	🟨	Left	mm	-0.02	0.00	1	0
Total Motor Power	IRB_2400_20kg_1.55m	Numeric	☐	🟦	Left	W	35.48	105.90	1	0

Figure 10. Graphic from RobotStudio *Signal Analyzer*. Total motor energy when moving the robot along the zigzag path located at the central location of the workspace [830, -100, 420], expressed in mm relative to the robot cs.



Signal Name	Source	Signal Type	Visible	Color	Axis	Unit	Min	Max	Scale	Offset
Speed In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP	Numeric	☐	⬛	Left	mm/s	0.00	100.02	1	0
Total Motor Energy	IRB_2400_20kg_1.55m	Numeric	✓	🟧	Left	J	0.0	4456.0	1	0
X Position In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP/Position	Numeric	☐	🟦	Left	mm	5.00	195.00	1	0
Y Position In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP/Position	Numeric	☐	🟩	Left	mm	5.00	95.00	1	0
Z Position In Current Wobj	IRB_2400_20kg_1.55m/Mechanical Units/ROB_1 [IRB2400_16_150__02]/TCP/Position	Numeric	☐	🟨	Left	mm	0.00	0.00	1	0
Total Motor Power	IRB_2400_20kg_1.55m	Numeric	☐	🟦	Left	W	108.61	312.64	1	0

Figure 11. Graphic from RobotStudio™ *Signal Analyzer*. Total motor energy when moving the robot along the zigzag path located at the boundary position of the workspace, [1530, -600, 420], expressed in mm relative to the robot cs.

V. CONCLUSIONS

The use of the software RobotStudio for teaching purposes is found to be particularly suitable for both educators and students. It provides the possibility to model the existing robotic cell and have a simulating environment that mimics the same procedures that are used with real robots. In this way, it is possible to let the students gain an insight in operating and programming an industrial robot not so effectively achieved by only theoretical classes. The use of the software was also instrumental in coping with the high number of students and with the lecturing time available. The lab assignment on jogging the robot using the virtual robotic cell is used to present joint and cartesian interpolation and to explore the differences between off-line and online procedures. With the virtual cell it is possible to let all students appreciate the natural difficulty that occurs when it is necessary to position the robot end effector in a given position and orientation using the FlexPendant. The second lab assignment on TCP calibration is used to illustrate the procedures that are carried out on on-line programming and used to introduce the mathematical calculation involved. The virtual cell offers the possibility for each student to perform this procedure in a similar way as in a real robotic cell. Students are able to obtain the TCP value of a gripper that is in line with the data of the model. This experience, although performed on a virtual cell, is still representative of the problems that are encountered in real practice. The third assignment is used to evaluate the robot path trajectory for different moving strategies to cover a plane surface, taking into consideration the time, the velocity profile along the path and the motor energy involved. It is found that the two strategies that involve only linear movements (zigzag and parallel spiral) are able to cover the surface in a shorter time that the true spiral strategy that uses linear and circular movements. Students are able to explore the path velocity profile for each strategy and verify that the true spiral presents a smoother velocity profile. In terms of motor energy used for each tested strategy, the differences are not relevant if it is taken into consideration the length of the path. One aspect that is verified is the influence on the location of the path within the robot workspace on the total motor energy. Moving the location of the zigzag strategy from the initial position (middle of the robot workspace) to a location in the same plane, but 500 mm apart, resulted in an increase on the total motor energy by a factor close to 3. This result draws the attention to consider the need for a careful choice on the location of the paths within the workspace of the robot, not only based on kinematic aspects but also on energy considerations.

The use of the virtual cell and assignments presented is perceived both by students and teachers as fundamental to learn and practice in spite of the reduced time and available resources that can be allocated to these lab sessions. The experience gained using this robotic virtual cell has been motivating us to further explore the use of virtual labs that close replicate the real system. The industrial software RobotStudio has proven to be a powerful tool for teaching purposes, keeping students motivated.

REFERENCES

[1] G. Bekey and J. Yuh, "The Status of Robotics," IEEE Robot. Automat. Mag., vol.15, pp.80-86, March, 2008. http://dx.doi.org/10.1109/M-RA.2007.907356

[2] S. Norf, Ed., *Handbook of Industrial Robotics*, 2nd ed., John Wiley & Sons, 1999.

[3] M. T. Restivo *et al.,* "A Remote Lab in Engineering Measurement," IEEE Trans. Ind. Electron., vol.56, pp. 4836-4843, December, 2009. http://dx.doi.org/10.1109/TIE.2008.2011479

[4] O. Cernian *et al.,* "The role of virtual laboratories in education," in *4th European Conf. E-COMM-LINE 2003*, Bucharest, 2003, pp. 135-140.

[5] C. Gravier *et al.*, "State of the art about remote laboratories paradigms – Foundations of ongoing mutations," Int. J. of Online Eng. (iJOE), vol. 4, n. 1, 2008.

[6] O. Goldstain *et al.*, "Remote learning for the manipulation and control of robotic cells," European J. of Eng. Educ., vol. 32, pp. 481-494, 2007. http://dx.doi.org/10.1080/03043790701337213

[7] C. A. Jara *et al.*, "An advanced interactive interface for robotics E-Learning," Int. J. of Online Eng. (iJOE), vol. 4, n. 4, 2008.

[8] C. A. Jara *et al.*, "Hands-on experience of undergraduate students in automatics and robotics using a virtual and remote laboratory," Comput. & Educ., vol. 57, pp. 2451-2461, 2011. http://dx.doi.org/10.1016/j.compedu.2011.07.003

[9] G. López-Nicolás *et al.*, "Active learning in robotics based on simulation tools," Comput. Appl. Eng. Educ., vol. 22, pp. 509-515, 2014. http://dx.doi.org/10.1002/cae.20576

[10] M. Cakir and E. Butun, "An educational tool for 6-DOF industrial robots with quaternion algebra," Comput. Appl. Eng. Educ., vol.15, pp. 143–154, 2007. http://dx.doi.org/10.1002/cae.20104

[11] S. Makris *et al.*, "Virtual Commissioning of an Assembly Cell with Cooperating Robots," Advances in Decision Sciences, vol. 2012, Article ID 428060, 2012. http://dx.doi.org/10.1155/2012/428060

[12] F. Gîrbacia *et al.*, "Off-line programming of industrial robots using co-located environments," Advanced Materials Research, vols. 463-464, pp. 1654-1657, 2012. http://dx.doi.org/10.4028/www.scientific.net/AMR.463-464.1654

[13] M. Sarkans and L. Roosimölder, "Implementation of robot welding cells using modular approach," Estonian J. of Eng., vol. 16, pp. 317–327, 2010. http://dx.doi.org/10.3176/eng.2010.4.07

[14] M. T. Restivo *et al.,* "Feeling Materials' Stiffness by Haptics for Training," J. of Materials Educ., vol. 36 (3-4), pp. 51-68, 2014.

[15] Z. Pan *et al.*, "Recent progress on programming methods for industrial robots," Robotics and Computer-Integrated Manufacturing, vol. 28 (2), pp. 87-94, April 2012. http://dx.doi.org/10.1016/j.rcim.2011.08.004

[16] ABB. RobotStudio^{TM}. Available: http://www.abb.com/robotics

[17] P. Abreu *et al.*, "Robotics virtual lab based on off-line robot programming software," in Experiment@ International Conference (exp.at'13), 2013 2nd, pp.109-113. http://dx.doi.org/10.1109/expat.2013.6703040

[18] Z. Gan *et al.,* "Calibration of a Robot Visual System," in *Visual Sensing and its Applications: Integration of Laser Sensors to Industrial* Robots, Z. Gan and Q. Tang, Eds. New York: Springer Science & Business Media, 2011, ch.4, pp.93-141. http://dx.doi.org/10.1007/978-3-642-18287-7_4

AUTHORS

P. Abreu has a Mechanical Engineering degree at Faculty of Engineering of University of Porto (FEUP) and, is with it since 1985. He has MSc from Cranfield Institute of Technology, UK, 1988 and, a PhD by the University of Bristol, UK, 1995. He is Assistant Professor, in the area of Automation, Instrumentation and Control, at the Mechanical Engineering Department, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal. (e-mail: pabreu@fe.up.pt).

M. R. Barbosa has a Mechanical Engineering degree at Faculty of Engineering of University of Porto (FEUP), 1986 and is with it since 1987. He has an MSc in Manufacturing Systems Engineering, 1990 and a PhD, 1998, both by the University of Warwick, UK. He is Assistant Professor, in the area of Automation, Instrumentation and Control, at the Mechanical Engineering Department, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal. (e-mail: mbarbosa@fe.up.pt).

A. M. Lopes received the PhD degree in Mechanical Engineering from the University of Porto, Portugal, in March 2000. He has been with the Faculty of Engineering of the same University since 1991. At present he is an Assistant Professor of the Department of Mechanical Engineering, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal. His research interests include complex systems, fractional calculus, nonlinear dynamics, simulation, robotics, and control. (e-mail: aml@fe.up.pt).

This work was supported by National Funds through FCT – Fundação para a Ciência e a Tecnologia, under the project UID/EMS/50022/2013. This article is an extended and modified version of a paper presented at the International Conference exp.at'13, held 18-20 September 2013, in Coimbra, Portugal. Submitted, 25 May 2015. Published as resubmitted by the authors 20 August 2015.