CAD-Based Off-Line Robot Programming

Pedro Neto, J. Norberto Pires
Department of Mechanical Engineering (CEMUC)
University of Coimbra
Coimbra, Portugal
(pedro.netoljnp)@robotics.dem.uc.pt

A. Paulo Moreira
Institute for Systems and Computer Engineering of Porto
University of Porto
Porto, Portugal
amoreira@fe.up.pt

***Abstract*—Traditional industrial robot programming, using the robot teach pendant, is a tedious and time-consuming task that requires technical expertise. Hence, new and more intuitive ways for people to interact with robots are required to make robot programming easier. The goal is to develop methodologies that help users to program a robot in an intuitive way, with a high-level of abstraction from the robot language. In this paper we present a CAD-based system to program a robot from a 3D CAD environment, allowing users with basic CAD skills to generate robot programs off-line, without stop robot production. This system works as a human-robot interface (HRI) where, through a relatively low cost and commercially available CAD package, the user is able to generate robot programs. The methods used to extract information from the CAD and techniques to treat/convert it into robot commands are presented. The effectiveness of the proposed method is proved through various experiments. The results showed that the system is easy to use and within minutes an untrained user can set up the system and generate a robot program for a specific task. Finally, the time spent in the robot programming task is compared with the time taken to perform the same task but using the robot teach pendant as interface.**

***Keywords*—CAD, robot programming, HRI, intuitive programming, industrial robot, off-line programming**

I. INTRODUCTION

Traditional manufacturing systems (often based on fixed automation) are being replaced by flexible and adjustable manufacturing systems. Due to its flexibility, programmability and efficiency, industrial robots are seen as a key element of modern flexible manufacturing systems. Nevertheless, there are still some problems that hinder the utilization of robots in industry, especially in the small and medium-sized enterprises (SMEs) [1]. Programming an industrial robot by the typical teaching method, through the use of the robot teach pendant, is still a tedious and time-consuming task that requires some technical expertise. In fact, manual teach methods are often time consuming and imprecise. Nonetheless, one of the biggest problems that SMEs are facing is the lack of skilled workers, especially experts in robot programming and at the same time in specific manufacturing processes, such as welding and painting. Therefore, new and more intuitive ways for people to interact with robots are required to make robot programming easier. The goal is to develop methodologies that help users to program a robot with a high-level of abstraction from the robot specific language. Another important factor is the ability to program a robot off-line, without stop robot production. Many different solutions have been proposed in literature to create intuitive human-robot interfaces (HRIs); through the development and implementation of user-friendly software interfaces dedicated to a specific industrial process [2]; using sensors attached to the human body to capture arm movements and thus teach the robot by performing gestures [3]; using vision-based interfaces [4]; and speech [5]. Since over the past few years, computer-aided design (CAD) packages are becoming more powerful and accessible, CAD-based solutions related to the HRI problem have been common (see section II). Notwithstanding the above, due to the specific characteristics of an industrial environment (reliability requirements) it remains difficult to apply such systems in industry as many systems have not yet reached industrial usage. Thereby, the teach pendant continues to be the common robot input device that gives access to all functionalities provided by the robot (jogging the manipulator, producing and editing programs, etc.). In the last few years, the robot manufacturers have made great efforts to make user-friendly teach pendants, implementing ergonomic design concepts, more intuitive user interfaces, color touch screens with graphical interfaces, a 3D joystick, a 6D mouse and developing a wireless teach pendant. Nevertheless, it is still difficult for an untrained worker to operate with a robot teach pendant. The teach pendants are not intuitive to use and require a lot of user experience, besides being big and heavy [6].

In this paper is presented a CAD-based system to program a robot from a 3D CAD environment, allowing users with basic CAD skills to generate robot programs off-line. In addition, the 3D CAD package (Autodesk Inventor) that interfaces with the user is a well known CAD package, widespread in the market at a relative low-cost. This system works as a HRI where, through the CAD interface, the user is able to generate robot programs. The methods used to extract information from the CAD (position and orientation of rigid bodies in space) and techniques to treat/convert it into robot commands will be presented. Experiments were conducted to evaluate the system performance. The results showed that the proposed system is easy to use and within minutes an untrained user can setup the system and generate a robot program for a specific task. The time spent in the robot programming task is compared with the time taken to perform the same task but using the robot teach pendant as interface. Experiments were performed with a six-axis industrial robot in laboratory environment, giving to the reader a good insight into the problem. Finally, results are

978-1-4244-6506-4/10/$26.00 ©2010 IEEE

discussed and some considerations about future work directions are made.

II. Related Work

In recent years, CAD technology has become economically attractive and easy to work with so that today millions of SMEs worldwide are using it to design and model their products. Already in the 80's, CAD was seen as a technology that could help in the development of robotics [7]. Since then, a variety of research has been conducted in the field of CAD-based robot planning and programming.

Many CAD-based systems have been proposed to assist people in the robot programming process. A series of studies have been conducted using CAD as an interface between robots and humans, for example, extracting motion information from a CAD data exchange format (DXF) file and converting it into robot commands [8]. In this system, the user only needs to define the welding path and the approach/escape paths in the drawing. Another study presents a method to generate three-dimensional robot working paths for a robotic adhesive spray system for shoe outsoles and uppers [9]. A robotic sanding platform where the robot paths are generated by CAD/CAM software is presented in [10]. Reference [11] presents a model based robot programming concept for applications where metal profiles are processed by robots and only a 2D geometrical representation of the workpiece is available. An example of a novel process that benefits from the robots and CAD versatility is the so-called incremental forming process of metal sheets. The robot's trajectories are calculated from the CAD model on the basis of specific material models. Prototype panels or customized car panels can be economically produced using this method [12]. Reference [13] presents a robot path generator for the polishing process, where the cutter location data is generated from the postprocessor of a CAD system. As we have seen above, a variety of research has been done in the area of CAD-based robot planning and programming. However, none of the studies so far deals with a "global" solution for this problem. Even though an abundance of approaches has been presented, a cost-effective standard solution has not been established yet. The system presented here allows to extract data not only from a single CAD model but essentially from a CAD environment representing a robotic cell. We can say that in this case, generate a robot program is simple as the manipulation of virtual parts in a 3D CAD environment.

III. CAD-Based Interface

In this paper, the information extracted from CAD models will be used to generate robot programs. Through the CAD interface, any user with basic CAD skills will be able to define the robot working paths and organise them in the desired sequence (definition and parameterisation of robot positions/orientations, reference frames, and trajectories). After completing the design, a program converts it into robot programs (off-line robot programming) (Fig. 1). The generated programs can be immediately tested for detailed tuning and a set of tools is then available to speed up corrections, if necessary. Depending on the complexity of the robotic cell, this process can be completed in just a few minutes, representing a huge reduction in programming and robot setup time.

Figure 1. Off-line robot programming concept. Working in an office environment, the user can generate robot programs without interrupt production.

IV. Proposed Approach

There is a lack of natural interfaces between humans and robots, something that allows us to show to the robot what it should do. As we know that one of the major challenges facing HRI involve the teaching of robots by operators, how we can interface/interact with robots in an intuitive way is the question. The HRI system should be intuitive, low-cost, with short learning curve, and should also allow users to program a robot in a short time.

Once CAD technology is widespread throughout the industry, we are proposing a CAD-based system to program an industrial robot, allowing users with basic CAD skills to generate robot programs off-line. The CAD package, Autodesk Inventor, will make the interface between the user and the robot. The information needed to program the robot will be extracted from the CAD models through an application programming interface (API) provided by Autodesk.

A. Application Programming Interface

The Autodesk Inventor API exposes the Inventor's functionalities in an object-oriented manner, allowing developers to interact with Autodesk Inventor using current programming languages (Visual Basic, Visual C#, Visual C++), for example, access to CAD data and create CAD models. There are different ways to access the API (Fig. 2) and it is important to choose the appropriate way to access it. In our system, a standalone application was used to extract information from the CAD and the Apprentice Server used to display the CAD models.

B. Method

The process begins with the extraction of data from the CAD, but first, it is necessary to specify what kind of data will be extracted and how. Robot programming is essentially based on the definition of robot paths, or rather, in the definition of a sequence of tag points that the robot passes through. As we are working with industrial robots, the tag points will define the robot end-effector pose, so that the definition of these points

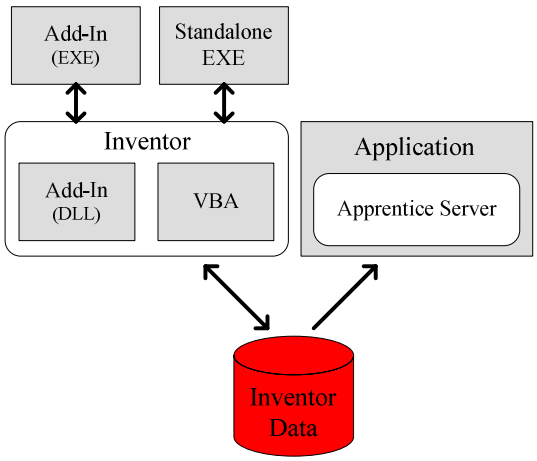

Figure 2. Different ways of accessing Autodesk Inventor's API. The white boxes represent components provided by the API (Autodesk Inventor and Apprentice Server) and the gray boxes represent programs written by developers.

should include not only position but also orientation in space. In our system the robot path is defined by a sequence of tool models that represent the end-effector pose in each point of the desired path. The API allows us to extract the transformation matrix of each part represented in a CAD assembly model. The transformation matrix of each part contains the rotation matrix and the position of the origin of that part, both in relation to the origin of the CAD assembly model. In addition, the API also gives us information about the position of the Autodesk Inventor WorkPoints, which are points that can be inserted in the CAD drawing at any location. Finally, the extracted information is converted into robot code.

C. *Position of Objects in Space*

In our proposed approach the tag points are defined by placing a WorkPoint in the tool model, where the tool is attached to the robot wrist (Fig. 3). The transformation matrix of each part (including tool models) is defined in relation to the origin of the CAD assembly model; however, the WorkPoints are defined in relation to the origin of each tool model in which they are attached. Therefore, considering our goal (definition of all the tag points in relation to a common reference), we need to define the WorkPoint coordinates in relation to the origin of the CAD assembly model or in relation to a reference frame defined by the user into the drawing.

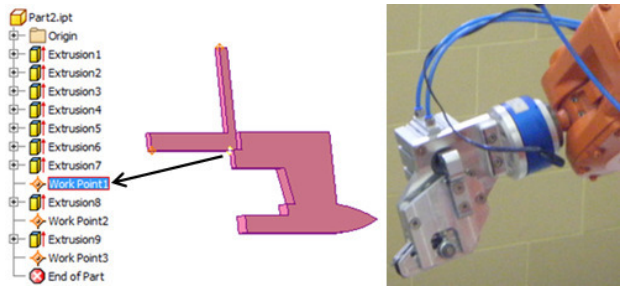

Figure 3. A simplified model of the robot tool (at left) and the real tool (at right). The tool can be modeled in a simplified manner but respecting its real dimensions, or rather, the important dimensions for the robotic process.

Considering two Cartesian coordinate systems; the first with origin and orientation coincident with the origin and orientation of the CAD assembly model (system A); and the second attached to the origin of a tool model (system B). Since the transformation matrix of each tool model is known we have the orientation and position of system B relative to system A. It is thus possible to make an analogy with Figure 4, where P is the WorkPoint, ${}^{B}\mathbf{P}$ represents the WorkPoint position in relation to the origin of the tool model, ${}^{A}\mathbf{P}_{\mathbf{Borg}}$ is the positional vector from the origin of the system A to the origin of the system B, and ${}^{A}\mathbf{P}$ is the positional vector of point P relative to the origin of the system A.

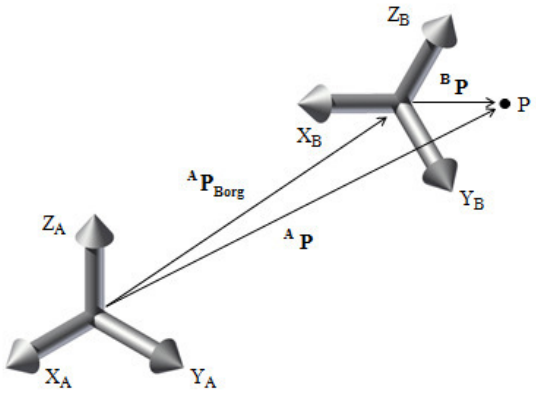

Figure 4. Reference systems A and B.

The aim is to obtain the vector ${}^{A}\mathbf{P}$. Considering the rotation matrix ${}^{A}_{B}\mathbf{R}$, which describes the orientation of the system B relative to the system A, we can write that:

$$ {}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R} \cdot {}^{B}\mathbf{P} + {}^{A}\mathbf{P}_{\mathbf{Borg}} \quad (1) $$

$$ {}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R} \cdot {}^{B}\mathbf{P} + {}^{A}\mathbf{P}_{\mathbf{Borg}} = {}^{A}_{B}\mathbf{T} \cdot {}^{B}\mathbf{P} \quad (2) $$

$$ \begin{bmatrix} {}^{A}\mathbf{P} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}_{B}\mathbf{R} & {}^{A}\mathbf{P}_{\mathbf{Borg}} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{B}\mathbf{P} \\ 1 \end{bmatrix} \Rightarrow {}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{T} \cdot {}^{B}\mathbf{P} \quad (3) $$

where the transformation matrix ${}^{A}_{B}\mathbf{T}$ and the vector ${}^{B}\mathbf{P}$ are known (provided by the API).

D. *Orientation of Objects in Space*

The transformation matrix of each tool in the CAD assembly model will be used to define the robot end-effector orientation in the form of quaternions (4), (5), (6), (7), where q_w is the real quaternion, and q_x, q_y, q_z are the imaginary quaternions. Since all the components t_{in} of the rotation matrix are provided by the API, the quaternions can be calculated.

$$ q_w = \frac{\sqrt{1 + t_{11} + t_{22} + t_{33}}}{2} \quad (4) $$

$$q_x = \frac{\sqrt{1 + t_{11} - t_{22} - t_{33}}}{2} \quad (5)$$

$$q_y = \frac{\sqrt{1 - t_{11} + t_{22} - t_{33}}}{2} \quad (6)$$

$$q_z = \frac{\sqrt{1 - t_{11} - t_{22} + t_{33}}}{2} \quad (7)$$

If the real quaternion q_w is nonzero, it becomes necessary to calculate the warning signs of the imaginary quaternions, given by (8), (9) and (10).

$$\text{sign}(q_x) = \text{sign}(q_x \cdot q_w) = \text{sign}(t_{32} - t_{23}) \quad (8)$$

$$\text{sign}(q_y) = \text{sign}(q_y \cdot q_w) = \text{sign}(t_{13} - t_{31}) \quad (9)$$

$$\text{sign}(q_z) = \text{sign}(q_z \cdot q_w) = \text{sign}(t_{21} - t_{12}) \quad (10)$$

E. Program Generation

The demand for intuitive ways to program machines has led to the emergence of techniques to generate code, such as the automatic generation of CNC code from CAD/CAM software. So why does not generate robot programs from CAD drawings? Using information from the CAD models, our system is able to generate control sequences for the robot program. In the construction of an algorithm to generate code, the keyword is "generalise" and never "particularise", in other words, the algorithm must be prepared to cover a wide range of variations in the process.

Several code generation techniques have been developed, but these tend to have drawbacks such as their suitability of the plans they produce for a particular application or how well they are able to generalise the problems. However, for a particular application with a limited and well known number of process variations, this kind of systems tend to present good performance. For this particular approach, the code generation process is divided into two distinct phases: first, definition and parameterization of robot positions/orientations, reference frames, tools, etc.; second, construction of the body of the program containing predominantly robot movement instructions (linear, joint, circular or spline robot movement type). The definition of the robot trajectories is an important part of the process, so from (3) we have the trajectory positions $\left({}^{A}\mathbf{P}\right)$ referring to a specific reference frame and from (4), (5), (6) and (7) we have the quaternions to define the orientation of the robot end-effector.

V. Experiments and Results

The CAD-based robotic system presented in this paper was designed to perform the task of object handling, or more specifically, the handling of knives in a robotic deburring cell. Briefly the system works as follows.

a) Create CAD model: The user should create a CAD assembly model representing the real robotic cell. The CAD model of a specific robotic cell only needs to be built once, so that usually the user only needs to make changes in design, according to the current work plan.

b) Place the robot tool models in the target positions.

c) Define robot parameters: Robot speed, robot tools, reference frames, etc.

d) Generate the robot program.

e) The generated program can be tested and adjusted if necessary.

A. System Architecture

The experimental cell is composed of an industrial robot (IRB 2400 equipped with the S4C controller, ABB), and a computer running the CAD package Autodesk Inventor and the software application that manages the cell (Fig. 5). The application receives data from the CAD, interprets the received data and generates robot programs. This application interface provides a set of capabilities to interact with the robot, allowing command the robot, upload and download programs to/from the robot, etc., using for this purpose the PcRob, an ActiveX created in our laboratory to control and manage the robot remotely (Fig. 6).

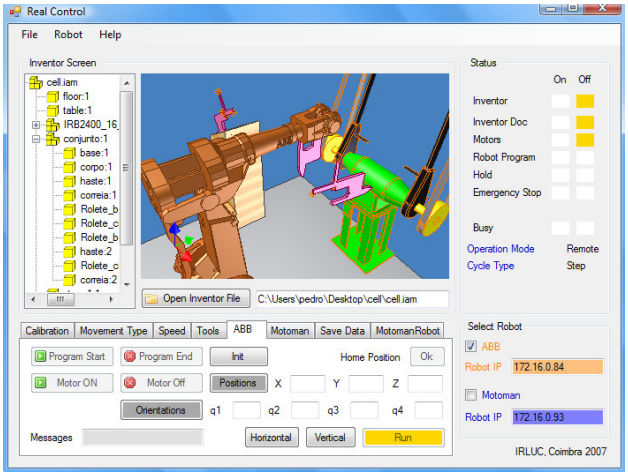

Figure 5. Software application interface.

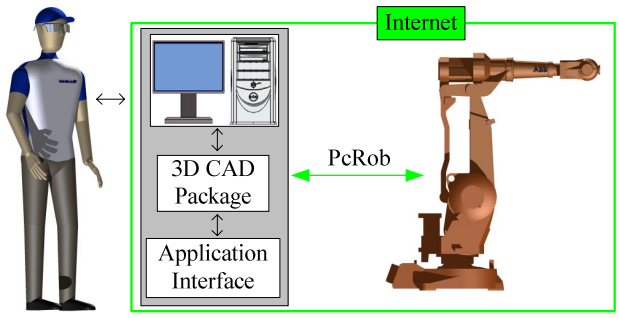

Figure 6. Communications and system architecture.

B. *Practical Implementation*

The CAD assembly model of the robotic cell (Fig. 7) does not need to accurately represent the real cell in all its aspects; on the contrary, it can be a simplified model. For example, the dimensions of the feed table, the relative position of objects (robot tools, feed table and deburring machine) and the robot tool length should represent the real scenario, however, the objects appearance need not be exactly equal to the real objects. When building the CAD model, it is important that the user keep in mind that the model will be used to generate a robot program. An important issue is that one must number the tool models according to the work flow because the algorithm to generate code is prepared to acquire data (transformation matrix and WorkPoints) from the first numbered tool to the last one. Another important issue is related to the process of calibration, specifically the placement of the reference frames. In this experiment we are using only one robot reference frame, and so a vertex of the feed table was selected to be the origin of the reference frame. The orientation for the reference frame is also defined into the CAD environment. This way, all positions from CAD are related to this reference frame.

After specifying some robot parameters such as robot speed, approach distances (the robot needs to reduce speed before placing the workpiece in the target location), etc., the system acquires data from CAD and convert it into a robot program. Finally, the generated robot program can be tested. It was generated a robot program to perform the task of object handling, or more specifically, the handling of unfinished knives in a robotic deburring cell. The robot starts by picking it up (from the feed table), moves it to a pose near to the deburring machine and finally puts it in contact with the deburring machine (Fig. 8). After this phase the process is controlled by a force control system [14]. In general, the generated program works very well. Results will be presented and discussed in the next section.

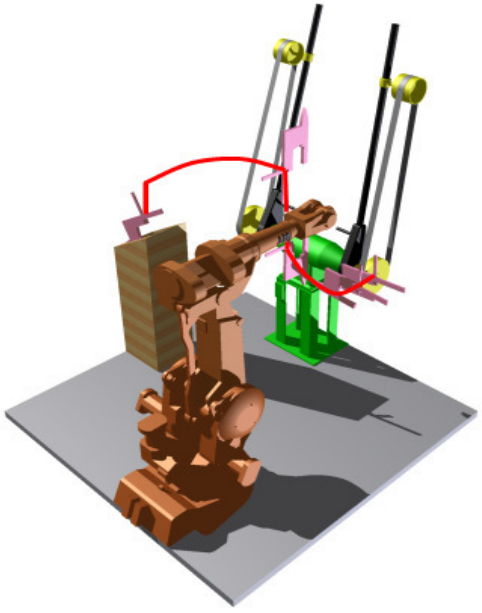

Figure 7. CAD assembly model of the robotic deburring cell. A robot program will be generated from this model.

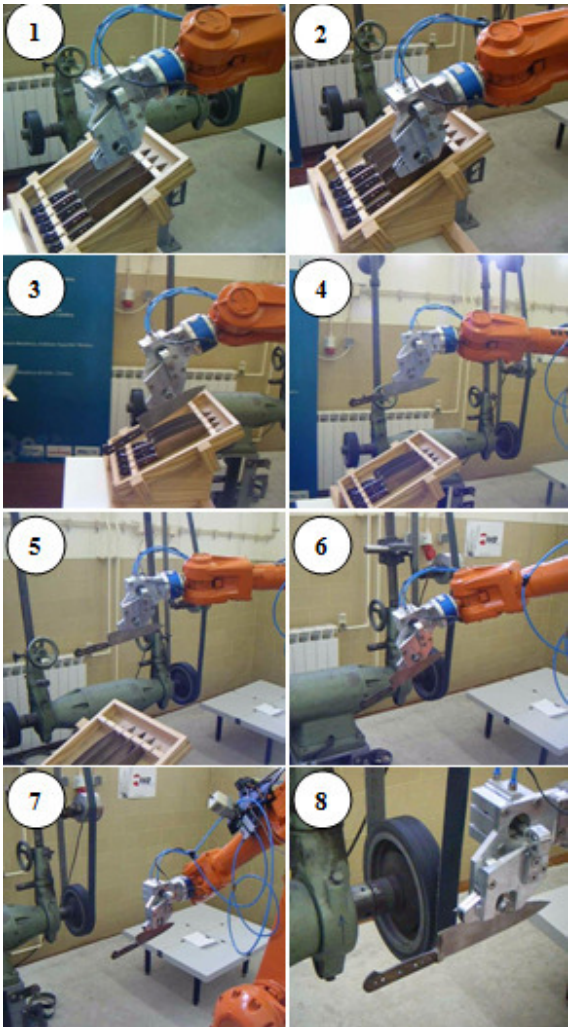

Figure 8. Robot running the generated program.

VI. Results, Discussion and Future Work

The experiments showed that the generated program works very well, without producing positional errors. The error that may exist comes from inaccuracies in the calibration process and from situations where the CAD model does not reproduce properly the real scenario. This is a problem that also affects the off-the-shelf off-line programming software's.

It is important to quantify the time spent in the process. Thus, the experiment above was conducted with two participants (P1 and P2), both with basic skills in CAD and 3D modelling, but had never worked with a robot before. After a brief explanation on how the system works (10 minutes) the participants began the test. This explanation was focused essentially on how the application interface works and the procedures to be followed at the time when the CAD model is constructed. The process was divided in five different tasks and the time spent in each one was reported in Table I. The construction of the CAD model consumes a great deal of time, on average about 64% of total time (robot and deburring machine models are provided by manufacturers), and if it was

considered that the acquisition of dimensions is part of the construction of the CAD model, then this task would consume over 85% of total time. Nevertheless, after the cell model is built, it can be quickly reconfigured to generate code for a different task in few minutes. Both participants generate the robot program and put it running on the robot controller with success, spending on average 33 minutes.

We thought it was interesting to compare the time taken to program the robot using this CAD-based system and using the robot teach pendant. The same participants (P1 and P2) were invited to conduct the experiment. In the first experiment (using the CAD interface) we spent 10 minutes explaining how the system works, but for the current experiment (using the teach pendant) we quickly concluded that this was not enough time, so we spent 2 hours talking about general robotics (a quick introduction) and 1 hour talking about robot programming using the teach pendant. This explanation focused mainly on practical aspects. Participant P1 spent about 58 minutes to complete the task (most of the time consulting the robot manual) while the participant P2 gave up. It can be concluded then that the first experiment took less time, however, we believe that with practice this 58 minutes may be reduced. The point to emphasize is that the time spent teaching participants was much lower for the first experiment (10 minutes) than for the second (3 hours), showing the short learning curve of the CAD-based system. Finally, when questioned, participants indicated that the first experiment (using CAD) is much more intuitive and therefore easy to use.

As future work, the way the system is calibrated should be simplified and less susceptible to errors. This will be done through a mechanism to on-line adjust the robot pose according to the error reported.

TABLE I. TIME SPENT BY PARTICIPANTS IN EACH TASK

	Time Spent (minutes)	
	P1	***P2***
Acquisition of real cell dimensions	7	7
CAD model construction	20	22
Definition of robot frame(s)	3	3
Robot and process parameterisation	1	1
Code generation	1	1
Total	32	34

VII. CONCLUSION

A new CAD-based system dedicated to off-line robot programming was developed. This system works as a HRI, allowing users with basic CAD skills to generate robot programs off-line from a CAD environment, and in an intuitive way. A common 3D CAD package (Autodesk Inventor) was selected to make the interface between user and robot. The effectiveness of the proposed system was proved through the experiments. These experiments were conducted with two participants that had never worked with a robot before. Experiments showed that the CAD-based system is intuitive and has a short learning curve, allowing non-experts in robot programming to generate robot programs in just few minutes. This low cost off-line robot programming technology can be an important tool for SMEs who have robots but do not have trained personnel to operate them.

ACKNOWLEDGMENT

This work was supported in part by the Portuguese Foundation for Science and Technology (FCT), grant no. SFRH/BD/39218/2007.

REFERENCES

[1] J. N. Pires, K. Nilsson, and H. G. Petersen, "Industrial robotics applications and industry-academia cooperation in Europe," IEEE Robotics & Automation Magazine, vol. 12, no. 3, pp. 5–6, 2005.

[2] J. N. Pires, "Semi-autonomous manufacturing systems: the role of the role human-machine interface software and of the manufacturing tracking software," Mechatronics: an International Journal, vol. 15, no. 10, pp. 1191–1205, 2005.

[3] P. Neto, J. N. Pires, and A. P. Moreira, "Accelerometer-based control of an industrial robotic arm," in 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2009), pp. 1192–1197, Toyama, Japan, 2009.

[4] U. Hillenbrand, B. Brunner, C. Borst and G. Hirzinger, "The robutler: a vision-controlled hand-arm system for manipulating botles and glasses," in Proceedings of the 35th International Symposium on robotics, Paris, France, 2004.

[5] S. Yamamoto, J. M. Valin, K. Nakadai, J. Rouat, F. Michaud, T. Ogata, and H. G. Okuno, "Enhanced robot speech recognition based on microphone array source separation and missing feature theory," in Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), pp. 1477–1482, Barcelona, Spain, 2005.

[6] B. Hein, M. Hensel, and H. Worn, "Intuitive and model-based on-line programming of industrial robots: a modular on-line programming environment," in Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008), pp. 3952–3957, Pasadena, USA, 2008.

[7] B. Bhanu, "CAD-based robot vision," IEEE Computer, vol. 20, no. 8, pp. 13–16, 1987.

[8] J. N. Pires, T. Godinho, and P. Ferreira, "CAD interface for automatic robot welding programming," Industrial Robot: an International Journal, vol. 31, no. 1, pp. 71–76, 2004.

[9] J. Y. Kim, "CAD-based automated robot programming in adhesive spray systems for shoe outsoles and uppers," Journal of Robotic Systems, vol. 21, pp. 625–634, 2004.

[10] F. Nagata, Y. Kusumoto, Y. Fujimoto, and K. Watanabe, "Robotic sanding system for new designed furniture with free-formed surface," Robotics and Computer-Integrated Manufacturing, vol. 23, no. 4, pp. 371–379, 2007.

[11] T. Pulkkinen, T. Heikkilä, M. Sallinen, S. Kivikunnas, and T. Salmi, "2D CAD based robot programming for processing metal profiles in short series manufacturing," in International Conference on Control, Automation and Systems (ICCAS 2008), pp. 156–162, Seoul, Korea, 2008.

[12] T. Schaefer, and R. D. Schraft, "Incremental sheet metal forming by industrial robots," Rapid Prototyping Journal, vol. 11, no. 5, pp. 278–286, 2005.

[13] L. Feng-yun, and L. Tian-sheng, "Development of a robot system for complex surfaces polishing based on CL data," The International Journal of Advanced Manufacturing Technology, vol. 26, pp. 1132–1137, 2005.

[14] J. N. Pires, G. Afonso, and N. Estrela, "CAD interface for automatic robot welding programming," Assembly Automation, vol. 27, no. 2, pp. 148–156, 2007.