International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:9, No:2, 2015

User-Friendly Task Creation Using a CAD Integrated Robotic System on a Real Workcell

Alireza Changizi, Arash Rezaei, Jamal Muhammad, Jyrki Latokartano, Minna Lanz

***Abstract*—**Offline programming (OLP) is a new method in robot programming which is used widely in the industry nowadays which is a simulation base method that can produce the robot codes for motion according to virtual world in the simulation software. In this project Delmia v5 is used as simulation software. First the work cell component was modelled by Catia v5 and all of them was imported to a process file in Delmia and placed roughly to form the virtual work cell. Then robot was added to the work cell from the Delmia library. Work cell was calibrated corresponding to real world work cell to have accurate code. Tool calibration is the first step of calibration scheme and then work cell equipment can be calibrated using 6 point calibration method. Finally generated code needs to be reformed to match related controller code instruction. At the last stage IO were set to accomplish robots cooperation and make their motion synchronized. The pros and cons also will be discussed to clarify the presented results show the feasibility of the method and its effect on production line efficiency. Finally the positive and negative points of the implementation will be discussed.

***Keywords*—**Component, robotic, automated, production, offline programming, CAD.

I. Introduction

IN order to have useful work done by a robotic manipulator, it must be programmed to accomplish the desired task or motion cycle. Nowadays industrial robots generally require a tremendous amount of programming to make them useful. Their controllers are very sophisticated [1]; the commercial robot programming environments are typically closed.

Systems and the programming languages vary among manufacturers. Despite the great evolution of the industrial robot controllers, in the majority of the industrial applications, the robot programming is made, using one of the following ways:

A. Manual On-Line Programming

On-line programming of the robots is a common method of robot programming which programmer requires access to the robot and teaches the points by jogging the robot manipulator using teach pendant and reach the targets or have the program automatic using sensors to predict the pass for gripper [4]. Then he needs to save the points in a program inside the controller and then defines velocity profile and motion profile for hitting the desired points. In this method programs exist only in the memory of robot control system often difficult to transfer, document, maintain, or modify. In the context of the online programming, usually a robot programmer hired by a factory must know specific languages of the robots employed in the factory floor. Also developing manufacturing process might probably need to integrate new brands of robot with different (or even unique) properties where the programming languages are different and it can cause extra fees to company to hire a new programmer or teach [5].

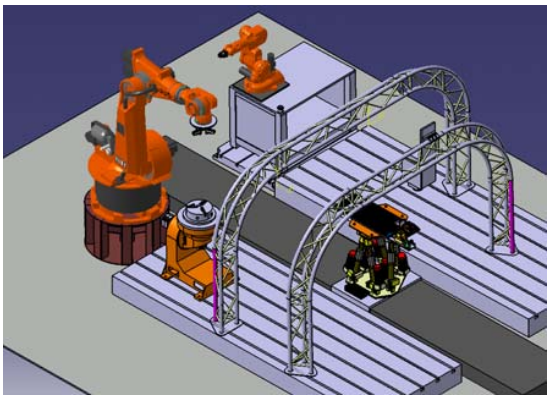

Fig. 1 Workcell in a 3D view

B. Off-Line Programming

Off-line programming (OLP) is a powerful tool for saving integrators and end-users time and money when designing a work cell. Different approaches can aim for material handling [3], automotive and industrial welding [2], digital manufacturing, etc. The ability to analyze how a work cell will behave before investing time and money on equipment makes for a smoother transition from concept to reality. Simulation and OLP allows integrators to study multiple scenarios of a work cell before any metal is committed. Mistakes commonly made in designing a work cell can be made in advance and studied. Simulation and OLP are also powerful teaching tools for engineering students.

In this context, purpose is to describe an offline programming of KUKA KR125 in heavy laser laboratory in Tampere University of Technology using Delmia v5.

A. Changizi is with the Tampere University of Technology, Department of Mechanical Engineering and Industrial Systems, P.O. Box 589, FI-33101 Tampere, Finland (phone: +358 40 198 1784; fax +358 3 364 1429; e-mail: alireza.changizi@tut.fi).

A. Rezaei and J. Muhammad were with Tampere University of Technology, Department of Mechanical Engineering and Industrial Systems, P.O. Box 589, FI-33101 Tampere, Finland (e-mail: arash.rezaii@gmail.com, jamal.muhammad.vc@gmail.com).

J. Latokartano and M. Lanz are with the Tampere University of Technology, Department of Mechanical Engineering and Industrial Systems, P.O. Box 589, FI-33101 Tampere, Finland (e-mail: jyrki.latokartano@tut.fi, Minna.Lanz@tut.fi).

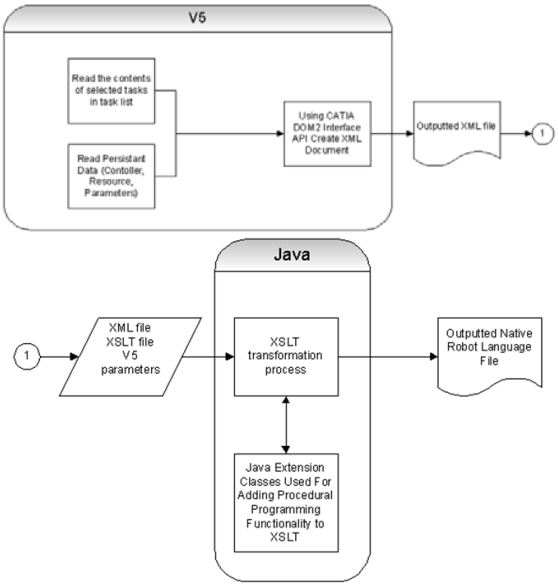

Fig. 2 Code conversion process

II. Modelling of Work Cell Equipment

At very first steps, all work cell equipment and work pieces was modelled in Catia V5 with an acceptable accuracy. Modelling accuracy influence the accuracy of the generated code for task markedly so more precise models, more precise code can be generated. To accomplish this task all the dimension was measured by tape and caliper, meanwhile relative distance of modelled objects was also measured to have a rough idea about placing the equipment in the virtual work cell in Delmia simulation environment. Fig. 1 shows a prepared model of the robots and the some of the equipment in the cell.

At first glance, the work cell looks regular and well-ordered with a good accuracy, whereas after plenty of measurements, it was revealed that there are many discrepancies and non-parallelity between symmetries component. Therefore a very accurate method is needed to precisely place all the equipment in the virtual world, correspondence to real word with all that irregularities in it. This will be considered during calibration process where all the distance and angle will be tuned according to data coming from real world. Moreover any dimensional error in 3D modelling causes inacceptable error in the generated code. Also designing all possible details can help user to have smothered and easier calibration process.

III. Process Simulation

The task of KUKA has been simulated in a process file under device task definition and off-line programming work benches. All the modelled part files in Catia were saved as product files and after save management were imported to a process file under resource layout workbench in the resource detailing category. Note that all equipment which needs to be calibrated should be imported as separate product file in the work cell. Robots can be added to the work cell from Delmia library under device task definition workbench. All well-known Robots brands which extensively are used in the edge industry (e.g. KUKA, ABB, MOTOMAN, FANUC, etc.) can be found in the library of the Delmia. Delmia V5 has many environments and quite wide range areas for different fields of robot manufacturing processes. Different processes such as arc welding, handling or grinding can be simulated in simulation environments in Delmia. Users are able to make some tasks and see the exact 3D graphical view of simulation. During virtually teaching the robot and editing tasks, it is possible to define different tool profiles and control the velocities and also choose many kinds of robot motion and so many other details.

All simulated task containing any task properties like call activity, tool profile, speed profile, IOs, etc. can be transform into robot language files under device task definition in created robot program. Delmia creates robot programs from the contents of a simulated task and related controller data and produce XML files as its general output. Each task related to any kind of robot would transform to a XML file. A java executor needed to be installed on the machine to transform the created XML files to that specific robot language using a java extension classes. Whole process is shown graphically in Fig. 2. There are some points one should notice. Check whether Java 2 SDK version 1.4.1 or later is installed on your machine. Check in case Java 1.5 or later is used then also xalan-j 2.7.1 must be installed. Note that translation process of java executor should be installed on the pc to translate from xml file (output of Delmia) to related robot language.

Any existing Robot program with all details can be loaded to the Delmia. Also a generated program in delmia can be downloaded and then copy to a robot controller. Generated files are compatible with corresponding robot and controller.

IV. Robot Calibration

The robot calibration has the most important influence for an accepted off-line programming because only if the virtual world can exactly be mapped onto the real world, the automatically computed programs can be used in practice. Calibration features in DELMIA products allow the user to identify the sources of position inaccuracy and to modify the simulation world to match the real world. This correction allows a generic simulation developed in DELMIA products to be downloaded to different work cells that are nominally identical but which differ slightly in the locations of their parts and devices, their tool offsets, and their robot signatures. The main problem is the tolerances of the robot. The entire deviations of the mechanics, e.g. elasticity of the gear and manufacturing inaccuracies are not considered since the measuring system of the robot measures the position of the rotor axis of the motors. While creating points in the virtual world, the effect appears as a real error. The necessary total accuracy e.g. in spot welding must be between 2 to 2.5 mm. regarding the robot however the absolute accuracy must be less than 1 mm.

V. Calibration Process

Calibration is an interactive process between real world and virtual world [6]. Fig. 3 show the possible frames in the environment used during calibration and simulation process. Delmia calibrate the positions and orientations of the work pieces and equipment in the work cell using recorded data by robot in the real world. For this, a spike shaped tool with a pointed edge need to be mounted on the robot to record the desired points. A spike shape which made during this project is shown in Fig. 4.

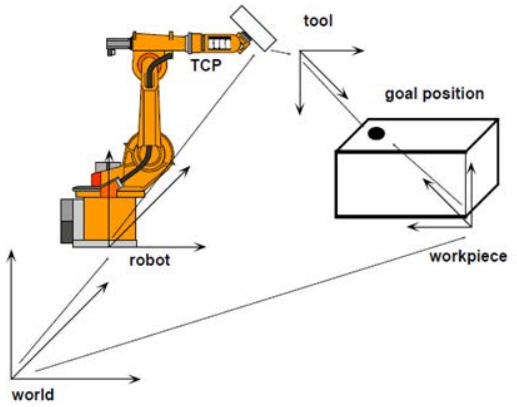

Fig. 3 Frames existing for all possible parts in the work environment

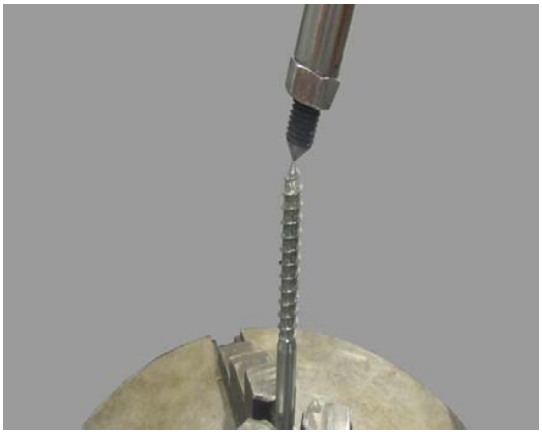

Fig. 4 Spike used for calibration of the rotational table

A. Tool Point Calibration

At first step it is necessary to calibrate spike. It can be done using the Tool Point Calibration option under OLP workbench in Delmia. The accuracy in determining tool profile of spike has the main influence on the accuracy of whole calibration process and accordingly the accuracy of whole work cell. Used spike is shown in Fig. 4.

A very sharp pointed spike can make the procedure easier and more accurate. The tool point calibration method adjusts the tool profile (Utool) of a robot based on the coordinates of mounting plate (0,0,0,0,0,0) of the robot for a fixed point in the space (as shown in Fig. 4) at different orientation of robot's wrist (robot configuration). Each orientation must be recorded in a point in the robot controller while the robot tool profile is (0,0,0,0,0,0), at least 6 orientation is needed. Each orientation will be represented by a tag after uploading the program to the Delmia. Next steps are inside the Delmia and it consists of selecting a robot resource and a tag group representing uploaded mount plate positions. The user also enters an initial guess for the tool point parameters. The guess parameters should be within a few centimeters of the actual tool parameters.

Delmia compute the tool profile according to given tag group and it generated a tool profile under robot tree in the software. Data related to this tool profile like relative position (x,y,z) of the spike relative to the TCP (6th joint of the robot) and corresponding orientation (a1,a2,a3) can be read from Delmia. These data need to be entered in the controller as new tool profile (spike) while measuring the points in the 6th point calibration process. The generated codes for 6 point calibration should have spike as tool profiles. This can be set in controller while measuring the point.

B. Six Point Calibration

Six point calibration deals with the main purpose of calibration, namely placing all component in the modelled work cell in simulation (Delmia work bench which deals with robotic process) according to real world data coming through a program which spike is used as it tool profile and 6th definite point with a special properties (further will be illustrated) are touched by pointed end of spike. Note that orientation is not important in this process; position is the only matter which should be considered. Hence the spike can touch the points in any agile. The main aspect in selecting the 6th point is consideration of the order in choosing the points on different planes. Each rectangular object in space contains 6 planes. Then 3 points in first plane and 2 in next plane and 1 in last plane is required and it needs to pay attention that all these 3 planes should be perpendicular to each other. Then operator should teach the points by the robot using spike. Then codes related to those 6 points by teaching the points will be given it to Delmia.

Note that if for any reason spike is unmouted from robot, tool calibration should be done again; because in remounting action different position and orientation may be achieved. Consequently for each tool calibration, a new tool profile should be created in controller and point should be taught by the new spike profile. Also the position of the selected point relative to the robot position should be considered. All points should be reachable and they must exist in work envelope of the robot. In some case due to large dimension of the part comparing to robot scale, an extension should be added to the part for achieving three perpendicular planes. Fig. 5 shows ABB table which was calibrated by KUKA.

After touching all six points, generated code can be downloaded to the Delmia using corresponded translator. This code consists of a task and a tag group named as the program in tag tree. In this step corresponding to 6 touched points, user should create 6 tag points on the component which is calibrated in this process. Then under OLP workbench, calibration panel,

6th point calibration button, user should select imported tag point related to touched point and then select created tags to coincide all 6 points. Now part is calibrated according to imported tags.

Fig. 5 Calibrated table using KUKA manipulator

Created tags related to teach points are shown in red in Fig. 6. Note that they are in three perpendicular plane to satisfy necessary condition for using 6th point calibration algorithm in Delmia kernel.

VI. IO Signals

Defining IO for gripping action and robot handshaking action is necessary. Actually the 3D simulation in software during work piece handling is only graphical and to change it to the real world, creating output signals is required. For this purpose one can create IOs in work cell sequencing environment. By attaching the signal to the target operation and putting it in true or false manner operator can also control the griper condition(open or close) and simultaneous operation of multiple robots in a work cell of real world. IO can help us to manipulate robot cooperation in real world. In fact every robot's controller has input and output module which is vital for a robot to effect surrounding environment. These signals can be a control signal to auxiliary equipment like a work positioner or a robot's rail or even pneumatic grippers. These are logical signals which can be dealt like Boolean variable in conventional plc programming. That worth to mention that some equipment such as grippers are normally closed or normally open; so a pulse of being true or false for short time e.g. 5 seconds can also be useful.

IOs are also important in cooperation of robots. Programmer can use this Boolean variable for robots conversation. Robots can perform their task sequentially using wait and set an IO. In the way that at end of each task when other robot should begin its task first robot can set an IO to be true and then second robot should begin its task with a wait for IO line. Finally if IO was set true, program can pass this line.

Fig. 7 illustrated how two robot controllers were connected in the work cell.

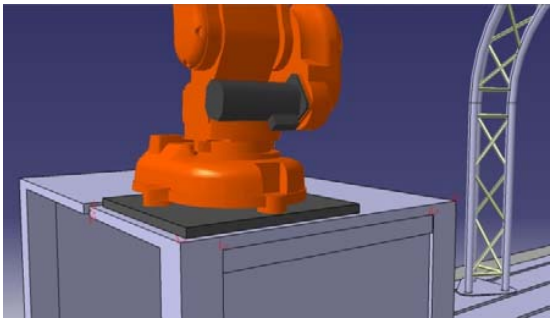

Fig. 6 Tags marked with red used for calibrating the table

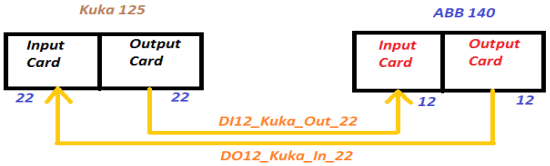

Fig. 7 IOs related ABB and KUKA robot connection

VII. Conclusions

The method was easy to implement and learning the process for workers with basic engineering education was about 1 to 2 hours including the software and robot implementation task creation. Simplicity of defining tasks in offline programming is a useful feature but it worth to mention at this point that during making it for real world, there will be no sensors installed on gripper or other parts; so collisions can happen related to many cases like errors in calibration, work piece modelling, positioning, or griper modelling.

Hence the important point is to be aware of these facts to avoid probable collisions.

To define tool profile, operator needs to consider the tool profile number in controller and in Delmia. The reason is that during importing the program through Java, the tool number is based on the Delmia, so if the tool defined in controller wouldn't have the same number, then the program will use some other profiles which can cause serious errors in implementation.

The other point is actually the tool point which is going to be used is not important. While using the same tool profile with same dimensions in software and robot controller, all to consider is the graphical simulation of the task that should be without collisions.

As far as the there is no extra sensor installed on the robots, there is the possibility for collisions or accidents in the workcell all the time. Marking the workcell space or installing fences around the robots is one method. As future work, it would be proposed to install cameras with object recognition modules to reduce the possibility for accidents.

References

[1] M. Bruccoleri, C. D'Onofrio and U. La Commare, "Off-line Programming and simulation for automatic robot control software generation," Industrial Informatics, 2007 5th IEEE International Conference on (Volume:1), pp. 491-496, June 2007.

[2] W. Dong, H. Li, X. Teng, "Off-line programming of Spot-weld Robot for Car-body in White Based on Robcad," Mechatronics and Automation, 2007. ICMA 2007. International Conference on, pp. 763-768, August 2007.
[3] P. Neto, "Off-line Programming and Simulation from CAD Drawings: Robot-Assisted Sheet Metal Bending," Annual Conference of the IEEE Industrial Electronics Society, IECON, pp. 4233-4238, 2013.
[4] P. Neto, "On-line automatic robot programming: A case study in grasping," Robotics and Automation. Proceedings. 1987 IEEE International Conference on (Volume:4), pp. 1292 - 1297, March 1987.
[5] S. Ishii, Y. MAEDA, "Programming of robots based on online computation of their swept volumes," Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on, pp. 385-390, August 2014.
[6] M. Abderrahim, A. Khamis, S. Garrido, L. Moreno, "Accuracy and Calibration Issues of Industrial Manipulators," Industrial Robotics: Programming, Simulation and Applicationl, ISBN 3-86611-286-6, pp. 702, ARS/plV, Germany, December 2006.