Virtual Engineering: Challenges and Solutions for Intuitive Offline Programming for Industrial Robot

Liwei Qi, Xingguo Yin, Haipeng Wang, Li Tao
ABB Corporate Research China
No. 31 Fu Te Dong San Rd., Waigaoqiao Free Trade Zone, 200131
Shanghai, P. R. China
Levy-liwei.qi@cn.abb.com，Xingguo.yin@cn.abb.com

***Abstract*—It's currently the market trend to make industrial robot easy to use. This paper analyzed the key areas and identified key bottlenecks that stop a robot simulation and offline programming tool to be intuitive and easy to use. The key areas are geometrical model handling, robot targets and paths handling, path configuration planning, etc. Based on the analysis, the solutions for each of the identified key areas were provided from easy of use perspective. The proposed solution has been implemented as a software product based on RobotStudio, ABB's simulation and offline programming tool.**

***Keywords*—industrial robot, cell, easy robot programming**

I. INTRODUCTION

The presented work related to robotic cell simulation and offline programming oriented to automated production applications. A robotic work cell includes a robot, one or more robot tools and one of more workstations that robot will visit and operate.

Conventional robot simulation and offline programming tools mainly oriented to skilled engineers rather than a person without too much engineering background, such as a sales man. To show a robotized solution via simulation to end customer directly from a sales man can guarantee fast response and better understanding of the proposed solution. From a sales man point of view, the existing simulation and offline programming tools are difficult to use from the aspects of modeling a geometrical entity; generating robot targets and paths; adjusting robot targets; planning path configuration; positioning cell objects; etc. In addition, to show different scenarios of the engineering solution for a customer always involves re-engineering, i.e., setting up virtual cell for simulation and offline programming from very beginning, which is time consuming, or even impossible regarding the engineering time.

In this paper, the key steps of setting up a virtual cell for offline programming and simulation are identified. For each of such steps, the challenges from easy of use perspective are analyzed. Based on the analysis, the corresponding solutions are presented with examples.

II. KEY STEPS TO SETUP A VIRTUAL ROBOTIC CELL FOR OFFLINE PROGRAMMING

Figure 1 shows the flow and key steps to setup a virtual robotic cell for offline programming and simulation.

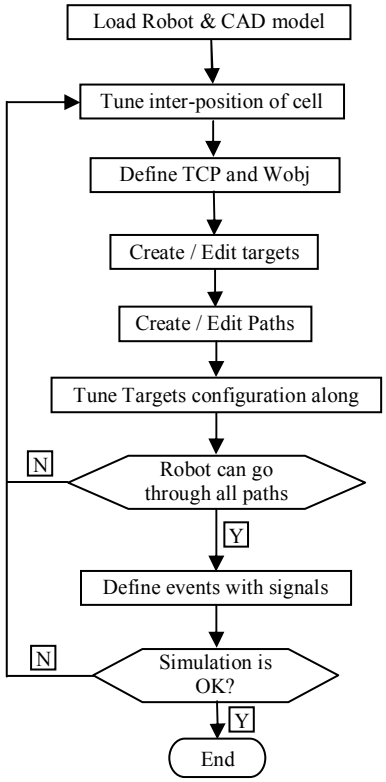

Figure 1. The key steps of offline robot programming and simulation

The starting point to create a virtual robotic cell is loading a robot model, and CAD models that form the cell as devices and facilities that a robot will handle. It is necessary to adjust the relative positions of robot and all devices as the initial cell layout. Based on this, the TCP – tool central point and the wobj – local reference coordinate system for different devices can be defined for later on robot targets creation. Robot targets needs then first created for different devices as the visiting points for a robot that can be linked as different robot paths. In order to make a robot go through all the paths within a cell, all robot targets need to be reachable for a robot. Based on this, the arm configuration of a robot when it visits a target needs to be correctly set to ensure robot can go from one target to another along all paths. Otherwise, either the positions of different cell components, includes different devices and the robot, or position & orientation of individual robot target need to be adjusted. When the needed events, such as pick a part, correctly defined with connection with different signals from either a

robot or a device, the offline program is ready and the simulation can be run. If the simulation result is satisfied from the perspective of robot motion or application needs, the offline programming is done. Otherwise any of the above steps need to be repeated for a good solution.

For the steps as shown in figure 1, good engineering background of geometrical modeling, robotics know-how, and usage of robot offline programming tools is necessary. How to lower the threshold of each of the key steps, or in another word, how to enable a person, such as a sales man without enough engineering background, to conduct offline programming and simulation is a challenging job.

III. BOTTLENECK ANALYSIS AND SOLUTIONS FOR KEY STEPS OF EASY OFFLINE PROGRAMMING AND SIMULATION

A. *Geometrical modeling*

The current way of geometrical modeling is either using a commercial CAD tool, such as Pro-Engineer or Solidworks, or the modeling module of a robot simulation software to build a complex CAD model with building blocks, such as box, cylinder. Several drawbacks can be identified for such a modeling way:

- The threshold to do so is high: at least several days training of the modeling tool is required and necessary practice is needed for an engineer;
- Geometrical modeling is a time consuming process even for a skilled engineer that prolong the preparation stage of offline robot programming
- In the solution proposal stage, once the size of a geometrical model needs to be changed, creating a new model from beginning is required.

In order to reduce the difficulty level and shorten the time for geometrical modeling, parametric driven modeling is presented. Based on the analysis of different robotic applications, the devices for a robotized cell can be categorized to form a device library. For each of the devices, the geometrical model can be standardized as one or several typical shapes. Such a geometrical model can be defined as a parametrical driven model. Figure 2 shows such a model for an injection molding machine.

In figure 2, the model of an injection molding machine is defined with 13 driven parameters. Each parameter has a default value to ensure the correct creating of a model. If the size of the model is not as desired, each of the parameter set can be changed separately or simultaneously to match the reality. In such a way a user can easily and quickly create a complex injection molding machine model without extensive knowledge of any traditional geometrical modeling tools. Besides that, with change of one or several driven parameters, a new model of an injection molding machine can be quickly created. Such a technology can be applied to any devices in a robotics cell that needs a 3D geometrical model.

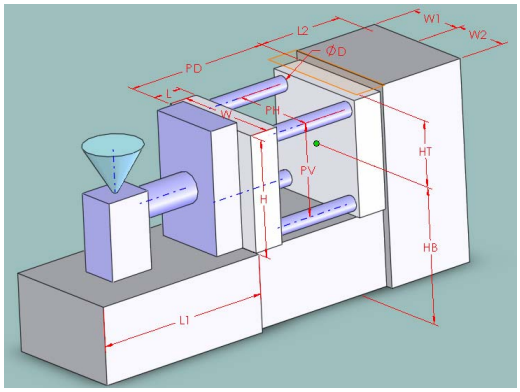

Figure 2. The parametric model of an injection molding machine

B. *Target and path generating*

The current way of target generating is either jogging a robot to a position to record the posture (position and orientation) of the tool center point, or input the value of the posture directly to define a robot target. Either way is not convenient and involves try-and-error regarding the precision of the created targets. Once all needed targets defined, it comes to the issue of how to organize different targets as one or several paths. This involves how to define the order of the robot motion, and how to define the properties of each robot motion item, such as motion speed, moving type, i.e., linear move or joint move, etc. This is also a time consuming process which demands extensive knowledge for a user regarding robotics knowledge and the skills to use a robot offline programming tool. Further more, even the above factors is not an issue, it is hard to merge the requirements of a specific robot application into the generated targets and paths, such as preferred posture of a target and motion setting of a path. This is highly depends on the experience of a user that can not guarantee a common standard.

In order to reduce the difficulty of robot target and path generating, and integrate well with the process requirements, an automatic robot target and path generating method based on parametric driven model is presented. The method is introduced based on but not limited to machine tending application.

Taking machine tending application as an example, there are typical devices, such as machines and conveyors, can be defined as different types of work stations that includes a geometrical model, two or more robot targets that formed as one or more paths. As introduced above, the basic shape of a parametric driven model is clearly defined with variation of the size. There is certain connection between the well defined geometrical model and the posture of a robot target based on the analysis of machine tending application. For example, the picking point for a molded part is always in line with the central point of the molding plate. To formulize such relationship between a robot target and the geometrical model as algorithms will enable the automatic way of target generating. Further more, the specific application needs on robot targets and paths can be organized as different templates. Such a template defines the order of targets in a path, motion properties of each motion items for a path, etc. With such a template, a robot path that is well matched with the application

needs can be quickly generated that can ensure a common standard of offline robot programming. Figure3 shows the relationship from geometrical model to robot targets and paths. In figure3, the model is the parametric driven model. Robot targets are created from the same set of the driven parameters for the model as shown in figure 2. All robot targets are organized as two paths (one in-path and one out-path) according to the process template.

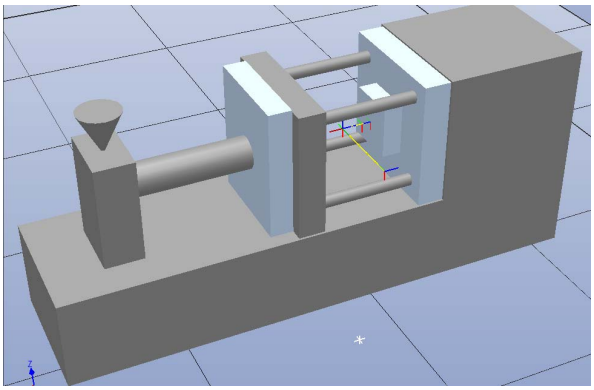

Figure 3. Figure3 A geometrical model and associated targets and paths

It is common that the size of a workstation needs to be changed to match with the needs of a specific application case. The proposed methods can avoid recreating a geometrical model from scratch as introduced above. Further more, since the targets are associated with a geometrical model, all targets belong to a work station can then be updated automatically to remain the same relationship with the station model. Hence an automatic way to generate and update a work station, includes geometrical model and all robot targets, is obtained.

C. *Scenario based robot programming*

It happens that different scenarios need to be evaluated and compared for a good solution. A scenario represents how a robot will tend a work station. For example one scenario for an injection molding machine represents picking a part from the front side of a machine, while another is for releasing a part to a machine, then picking a new molded part from the top of the machine. A scenario is formed with different actions for robot motion, tool methods and station logics. Robot motion involves robot paths, tool method involves tool data and tool signal logics, and station logic encapsulates station signal handling. Station logic is not supposed to be discussed here. Different templates need to be provided to match with different scenarios. But different scenarios of a work station are all based on the same station model.

In order to create different scenarios, the traditional way is to copy the station model and to create another set of the robot targets and paths manually, plus different tool methods and station logic setting. This is quite time consuming operation. The method introduced in this paper, i.e., scenario based robot programming, can not only realize automatic generating of geometrical model, targets and paths, but it is possible to map one geometrical model with different set of robot targets and paths from different scenarios. This makes it quite easier and faster to switch and compare different layouts, i.e., scenarios, of a specific application case.

In figure4 two scenarios of an injection molding machine is displayed. Scenario-1 represents picking a part from an injection molding machine, while scenario-2 represents putting an insert part into the same work station and then going to another side of the used mould to pick a molded part.

Once a new scenario is chosen for a work station, the actions collection will be changed according to the new scenario template. That means robot will use different paths when travels inside a station, use different tool methods to handle part with new station logic. From then on, different path templates can be used for the same geometrical model of a work station.

Scenario-1

MoveIn (Path1);

PickPart (Part1, doTool1Closed, 1, Tool1, 0.5);

MoveOut(Path2);

Scenario-2

MoveIn (Path3);

ReleasePart (Insert1, doTool2Closed, 1, Tool2, 0.5);

InsideTravel (Path4);

PickPart (Part1, doTool1Closed, 1, Tool1, 1.0);

MoveOut(Path2);

Figure 4. Two scenarios for an injection molding machine station.

D. *Target configuration planning*

It is usually possible to attain the same robot target in several different ways, using different sets of axis angles. These are called as different robot configurations[1]. Figure5 shows two different arm configurations used to attain the same robot target. The configuration on the right side is attained by rotating the arm backward, and axis1 is rotated 180 degrees.

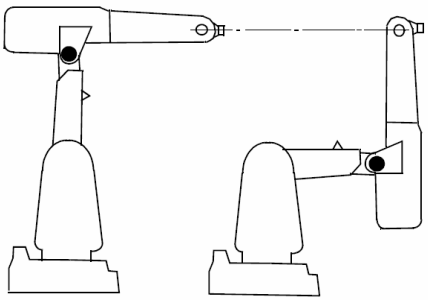

Figure 5. Two different arm configurations when robot approaches a target

Robot can reach two targets does not mean it can go along the path between the targets. This depends on if the rotation limits of any axis reached when robot moving. Since in most cases there are more than one configuration option for one robot target, this raises an issue of how to choose an appropriate configuration data for each robot target to ensure robot going along an entire path. Further more, the configuration setting for each robot target should ensure a smooth robot motion for a cycle that serves as the collection of paths for a robotic cell.

Currently for most of the offline robot programming tools, the configuration data for each robot target is defined by a user manually. This is again a time consuming process which has a high usage threshold, and highly depends on the knowledge and the experience of a user. For RobotStudio, the offline programming and simulation tool from ABB, it is possible to do target configuration planning for a path automatically. This is a big improvement compared with other similar tools. But how to set the configuration data for the starting and ending targets of a path to ensure a smooth change from one path to another along a robot motion cycle is still open. The methods presented here is for targets configuration planning along the whole robot motion cycle.

The principle, instead of the detailed implementation, of the targets configuration planning algorithm is introduced. The flow of targets configuration planning algorithm is shown in figure6. In figure6, only the configuration data selecting for two adjacent targets are shown to explain the principle.

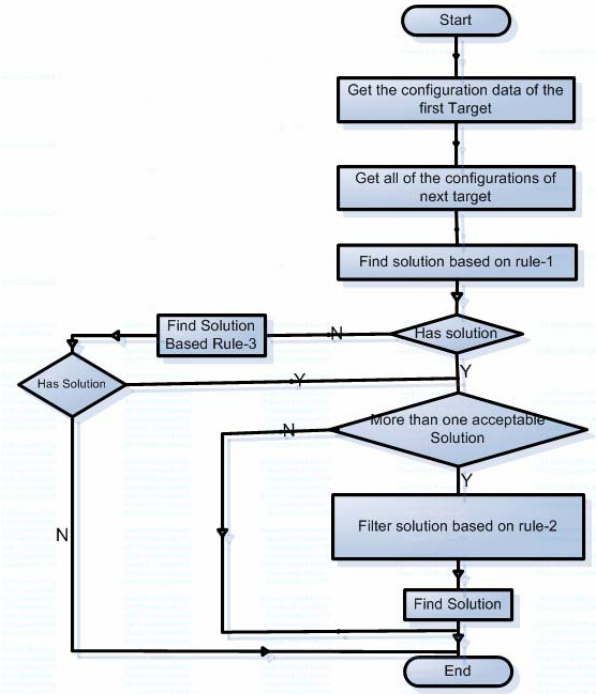

Figure 6. Algorithm of target configuration planning

For the algorithm in figure6, the key factor is to define the criteria for configuration selection for the later target based on the configuration data of the former target. A tight rule, called rule-1 is applied first. **Rule-1**: the selected configuration for a target should ensure any of the 6 axis of a robot not rotate more than 90 degree when the robot comes from the previous target. If more than one solution of the configuration for a target can be found based on the rule-1, another rule, rule-2, should be applied. **Rule-2**: to calculate the sum of the joint angle change from one target to another based on equation (1), choose the configuration data that can realize minimum distance in joint space. If there is no solution based on Rule-1, a loose rule called Rule-3 should apply. **Rule-3**: the selected configuration for a target should ensure any of the 6 axis of a robot not rotate more than 180 degree when the robot comes from the previous target. But it can happen that there is a solution according to Rule-3, but any of the robot axis can be out of reachable range when robot start to move based on such a solution. A test run should be conducted to finally verify if the solution is ok or not.

$$Dis = \sum_{i=1}^{6} \left| \alpha_{i,j} - \alpha_{i,j-1} \right| \tag{1}$$

Where in:

i is the number of the robot joints;

j is the current target No.

E. Cell components placement

How to placement all components of a robotic cell is very important in terms of making a good use of the capacity of a robot and improving productivity of a robotic cell. To conduct the job of cell components positioning needs a user has good robotics knowledge and rich experience of the robotic application which again not possible for a sales man. The solution for optimal cell components placement in terms of minimum robot cycle time is presented in another dedicated paper [2]. Here only a brief introduction of the solution is given as a part of the total solution. The flow the cell components placement is shown in figure7.

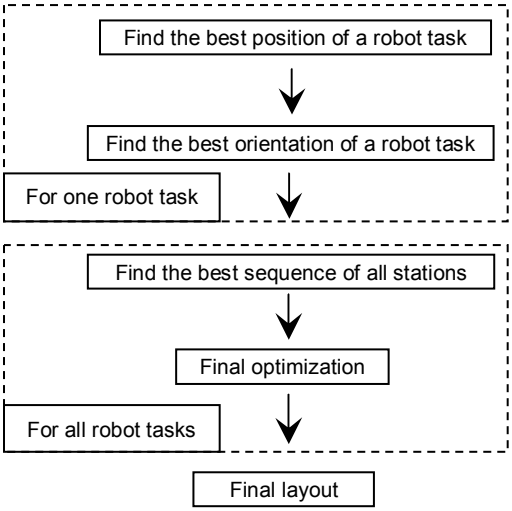

Figure 7. Flowchart of optimal cell placement

Usually there is not only one robot task in a robotic cell. An assumption for robot task positioning is applied for the presented work: repositioning of a robot task will not result in difference regarding robot performance if such repositioning will only cause different rotation of the first axis of a robot. This assumption has been verified with test cases. Based on the assumption, the positioning of several robot tasks can be divided into 3 stages, i.e., individual task positioning; task sorting; and optimizing. Firstly, each individual robot task will be considered separately for a best position and orientation relative to a robot. Secondly, based on the visiting sequence of all robot tasks, there should be a best order of all involved robot tasks, i.e., how the neighboring relation will be of all involved robot tasks to reach minimum cycle time. The first two stages can be regards as the initialization for the final optimizing. In

the final optimizing stage, position and orientation of all robot tasks can be adjusted simultaneously with use of the simulated annealing algorithm. As the result, the optimal robot tasks layout is better compared with that from optimal robot positioning [3]

IV. EXAMPLES

As the total solution for easy offline robot programming, the work presented here has been implemented as a software product called as Machine Tending ProcessPac (MTPP) tailed to RobotStudio, ABB's offline programming and simulation platform. Examples for each of the key areas discussed above are provided.

With MTPP, an IRB1600 robot, together with two work stations, one robot tool, one robot base and one cell fence can be quickly configured to form a machine tending robotic cell, as shown in figure 8. One work station is an injection molding machine, another is a conveyor. The robot will first go into the machine to pick a molded part and put it onto the conveyor to form a robot task cycle. Model of both stations, including the robot base and cell fence, are parametric model whose size can be easily changed to match with reality. Such changes on model size can be reflected in figure9.

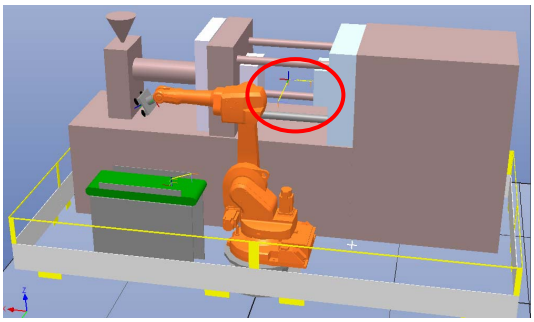

Figure 8. Sample machine tending robotic cell

Targets are created from the same set of driven parameters of both stations are displayed in figure8. These targets further formed as different paths according to the scenario that defined based on application requirements. The scenario of a station can be easily changed according to reality. For example, the scenario of the injection molding machine is changed from "pick from back side" to "release and pick from top", which means robot will first go into the machine to release a part (usually an insert) at one side of the plate, and then go to another side of the plate to pick a molded part and comes out of the machine from its top side. The predefined scenarios for an injection molding machine are shown in figure10.

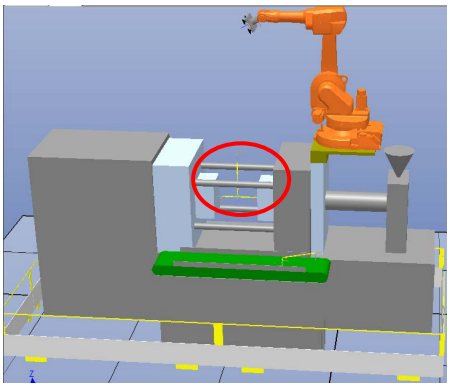

Figure 9. The same machine tending cell with modified cell configure data

If the scenario chosen for the injection molding machine station is "release and pick from top" as shown in figure10, the corresponding robot executable program generated is shown in figure11. In figure11, there are some actions for signal handling that do not discussed in this paper. For the path action, such as **ExecutePath "InReT_1"**, the executable robot program generated is shown in the same picture.

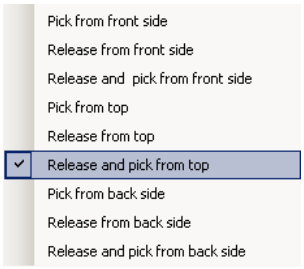

Figure 10. Different scenarios of an injection molding machine

```
28  PROC ES180_1()
29      WaitInput "diMouldOpenPos","1";
30      SetOutput "doMouldClose","0";
31      ExecutePath "InReT_1";
32      ReleasePart "ParaPart_1","doToolClosed","0",0.5;
33      ExecutePath "RePickT_1";
34      PickPart "ParaPart_1","doToolClosed","1",0.5;
35      ExecutePath "OutPickT_1";
36      SetOutput "doMouldClose","1";
37  ENDPROC
38
39
40  PROC InReT_1()
41      MoveJ Top_1,vmax,z50,VacuTool_1\WObj:=ES180_1_WO;
42      MoveJ MiddleTop_1,v3000,z100,VacuTool_1\WObj:=ES180_1_WO;
43      MoveJ RNearWork_1,v2000,z10,VacuTool_1\WObj:=ES180_1_WO;
44      MoveL Right_1,v200,fine,VacuTool_1\WObj:=ES180_1_WO;
45  ENDPROC
```

Figure 11. The robot program generated for the selected scenario

All the work stations, as the place holder for different robot tasks, are positioned automatically with the method introduced in the cell components placement section for an optimal cell layout in terms of minimum robot executing cycle time.

V. SUMMARY

The presented work analyzed the key challenges of easy offline robot programming. For each of the identified bottleneck, the corresponding solutions were provided. The solutions have been implemented as a software product called

as MTPP that based on RobotStudio. Together with other solutions in MTPP that can not be introduced limited by the paper size, a total solution for easy offline robot programming and simulation can be realized. The solution enables a sales man to create a virtual robotic cell and hence run simulation directly within an hour.

A solution engineer can work out detailed solution based on the results from a sales engineer. Or in another scenario, MTPP can be used directly by a solution engineer for offline programming purpose.

REFERENCES

[1] ABB LTD, RAPID Reference Manual 4.0.40. 2002.

[2] Zhang, D., Qi, L. Virtual Engineering: "Optimal cell layout method for improving productivity for industrial robot", IEEE CIS & RAM 2008 (Not published).

[3] ABB RESEARCH LTD, "METHOD FOR OPTIMISING THE PERFORMANCE OF A ROBOT", Patent Number: US 2007106421, 2007.