Automation of robot programming in milling

P Kozlov, S Mikhel and A Klimchik
Center for Technologies in Robotics and Mechatronics Components, Innopolis University, Innopolis, Russia

p.kozlov@innopolis.ru

Abstract. The paper demonstrates a method of automatic robot programming on the example of milling. The following problems are considered: format of the robot commands, analyzing an obtained trajectory, translation into the robot-specific language. The results confirm the ability of practical application for the proposed technique.

1. Introduction
Automatic programming can be used to simplify the application of the industrial manipulators for common operations [1]. The result of such programming is typically a script file. This script differs for every manufacturer of industrial manipulators. Sometimes it differs for manipulators even in one company, for example, KUKA IIWA requires a different programming method than other KUKA models. At the same time, the number of operations that an industrial manipulator has to do is limited to a small set of commands [2]. Therefore the algorithm of processing can be described in a robot-independent form and then translated to the specific platform.

Some developers introduce their languages to describe the robot operations [3, 4]. This makes sense when complex or collaborative tasks have to be done. But for most of the monotonic industrial operations, it is enough to define the sequence of commands that can be translated into the robot source code almost one-by-one. In this paper, we are going to describe such a system for the automatic programming of the milling process.

2. Materials and research methods
Let us look at the process of generating a control program on the example of milling. It contains the following steps.

- Find an area to process. In the case of milling, one can subtract the approximated model and initial model. It provides mesh difference that should be milled.
- Create a milling sequence. Generally, the milling process will use linear, spline, and circular movements.
- Configure translator using the manipulator parameters. Different manipulators have unique geometrical parameters that must be taken into account in addition to the robot language syntax.
- Translate the result into the robot specific language. It produces a program that can be easily uploaded to the robot.
The milling sequence can be saved independently from the generated robotic script. It allows regeneration without creation milling sequence and hence it will reduce production time. Milling sequence stores two arrays: points and operations. First one, contains all point (table 1) which the robot end-effector will attend. Sometimes end-effector may attend the same coordinates several times. Hence storing coordinates as variables will save a bit of memory. Second one, contains all operations (table 2) with information about requested points for current operation. The manipulator moves sequentially from one point to another, it means that the last point of the robot state in the previous operation is the first point of the new motion, and only one (last) position has to be defined. The exception is circular motion where one intermediate point must be set to define the shape of the curve. Parameter "proc" stores information about the milling process. It indicates is there any contact with the material, that can be used for decision making. For example, if the robot is not in contact, it can move faster or turn off the instrument.

Table 1. Robot end-effector points.

| Name | X   | Y   | Z   |
|------|-----|-----|-----|
| p0   | 0   | 0   | 0   |
| p1   | 15  | 15  | 50  |
| p2   | 15  | 15  | 45  |
| p3   | 35  | 15  | 45  |
| p4   | 35  | 15  | 50  |
| p5   | 15  | 20  | 50  |

Table 2. Robot generated commands.

| Type | Proc | Points |
|------|------|--------|
| ptp  | false| p0     |
| lin  | false| p1     |
| lin  | true | p2     |
| lin  | true | p3     |
| lin  | false| p4     |
| lin  | false| p5     |
| lin  | true | p3     |
| lin  | true | p4     |

In general, 3 parameters of the end-effector orientation have to be defined. For example, it can be roll-pitch-yaw angles. Most of the time robot moves with a constant orientation of the end-effector, and it is practical to simplify the trajectory representation via skipping the orientation parameters like in table 1.

The list of points is generated in the frame of workpiece because the generator does not have any information about the robot to be used, it deals with the tool motions only. In practice, the workpiece can be located in any part of the robot workspace and has an arbitrary orientation. Therefore it should be transformed into the most convenient frame for the current task. For example, if we are going to use the robot base frame, and the workpiece has displacement $P$ and rotation $R$ with respect to the robot, then each $i$-th point of the workpiece $p_w$ must be translated into the robot frame $p_r$ via transformation $p_r^i = Rp_w^i + P$. 


The found points can be translated into the robot program directly. It is the simplest approach that does not require any information about robot kinematics. However, some points can be out of the robot workspace, or the path can pass through a singular state. Therefore the found trajectory should be verified previously. Such verification requires solving the inverse kinematics problem. We used the Jacobian-based technique [5]. The method is selected for the following reasons:

- it is universal and can be used for different manipulator structures;
- it allows to find singularity, for example, from the manipulability parameter \( \sqrt{\text{det}(JJ^T)} \), where \( J \) is the manipulator Jacobian matrix, \( \text{det}(\cdot) \) is the determinant [6].

In general, the Jacobian based approach does not allow to find multiple solutions, but in the case of milling the sequential points are typically close to each other and the manipulator has to move smoothly, then it is enough to have the closest solution only. If the singularity is found, it can be avoided using the control in joint space, in this case, the "lin" and "circ" commands have to be transformed into the equal "ptp" form. If the trajectory cannot be executed, the operator sees a warning message before starting execution on the robot. Such an approach minimizes the possibility to have an error during real workpiece processing. It should be noted that in the process of verification the tool transformation must be taken into account.

The final step is generating the robot script. We used Python for the following reasons:

- powerful linear algebra library for working with Jacobians and kinematics;
- xml library for parsing the Universal Robot Description Format (URDF);
- simplicity of working with data collections and strings.

The translator works as follows. It takes two files as command-line parameters: URDF-file with the robot kinematics and the obtained trajectory. Usage of the URDF file allows us to easily change the format of the generated program and the verification settings. It is used in the constructor of the class "Robot" that reads kinematic parameters of the manipulator and uses it when the methods "FK" (forward kinematics) or "Jacobian" are called. The points and commands of the trajectory are represented in such a way that can be directly translated into the Python lists and dictionaries which allows us to avoid the explicit data parsing. Then the robot model virtually moves from one point to another and check its reachability and the robot manipulability. If there are no problems, the commands can be translated into the robot language format. Typically it requires us to define some obligatory instructions in the beginning and the end of the file, in particular, to wrap the control program into the procedure or function.

Example of the generated program is shown in table 3. It contains a fragment of the generated ls-file for a Fanuk manipulator, that corresponds to the sequence of instructions in table 2. Each command contains several parameters additionally to the point index, such as velocity, acceleration, and other motion-specific elements. In our implementation they were defined in translator, but in general they should be generated together with the path points.

3. Results and discussion

This approach provides a possibility to execute one algorithm on several and sometimes different industrial manipulators. It only requests an industrial manipulator itself, generated program, and the translator.

The described conversion is also applicable to the other stages. For example, it is possible to create a program for industrial manipulators for assembling complex workpieces from typical components. The assembling is also included an automated glue application for desired sides. Thus, a laptop is only requested during model approximation and program generation. It produces the opportunity to use the environment more efficiently.
**Table 3.** Output code example.

| Type | Point | Parameters                           |
|------|-------|--------------------------------------|
| 4:J  | P[1]  | 10% FINE ACC10;                      |
| 5:L  | P[2]  | 500 cm min⁻¹ CNT80 ACC65 ;           |
| 6:L  | P[3]  | 500 cm min⁻¹ CNT80 ACC65 ;           |
| 7:L  | P[4]  | 500 cm min⁻¹ CNT80 ACC65 ;           |
| 8:L  | P[5]  | 500 cm min⁻¹ CNT80 ACC65 ;           |
| 9:L  | P[6]  | 500 cm min⁻¹ CNT80 ACC65 ;           |
| 10:L | P[4]  | 500 cm min⁻¹ CNT80 ACC65 ;           |
| 11:L | P[5]  | 500 cm min⁻¹ CNT80 ACC65 ;           |

Program generation for building complex workpiece uses several moving types. Moving types might be in joint space or linear (Cartesian) space. Linear space motion is using when there is a requirement to move linearly. Also, there might be some environment restrictions for example limited environment or finally placed components. Joint space motion is not linear in Cartesian space but it provides faster manipulation.

During workpiece production linear moving is required for:

- Typical component picking (figure 1 (a));
- Glue application;
- Component final positng with force applying (figure 1 (b)).

The joint space motion is used during moving between component picking and placing operations. It is allowable because there is no requirements on the accuracy or any other environment limitations.

![Image](image_url)

(a) Picking typical component  
(b) Placing typical component

**Figure 1.** Building complex workpiece demonstration.

The limitation of the proposed approach is its focus on sequence of predefined motions, therefore it cannot be used in cases, when the robot has to choose different motions depending on the sensory information or other factors. It can be solved by increasing the number of commands and complexity of translator, but in this case introduction of an intermediate language instead of the command set looks more advisable.

4. **Conclusion**

The paper demonstrates the method of robot programming, based on the automatic trajectory generation and process description in terms of basic motion commands in robot-independent
form. This approach was applied for generating script for milling on the Fanuc and KUKA industrial manipulators. The results confirm the possibility of its practical application. The proposed technique can be used in other robot tasks, such as assembling operations.

In future work we are going to improve the trajectory generator to estimate all the required parameters of the process during the path planning and remove this part from the translator. Also we are going to introduce a variable angle for the milling process. It can be achieved because there is some concave parts which cannot be milled by horizontal configuration. It will produce a possibility to increase an accuracy for the final model.

References
[1] Mitsi S et al 2005 Off-line programming of an industrial robot for manufacturing The Int. J. of Advanced Manufacturing Technology
[2] Nilsson K 1996 Industrial robot programming Department of Automatic Control, Lund Institute of Technology
[3] Scripts in SprutCAM Available at: https://kb.sprutcam.com
[4] RCML manual Available at: http://robotct.com
[5] Buss S R 2004 Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods IEEE J. of Robotics and Automation
[6] Wang S L and Waldron K J 1987 A study of the singular configurations of serial manipulators