Proceedings of the 2015
IEEE Conference on Robotics and Biomimetics
Zhuhai, China, December 6-9, 2015

A Robotic Off-line Programming System Based on SolidWorks

Hongmin Wu, Huajian Deng, Cao Yang, Yisheng Guan*, Hong Zhang and Hao Li

Abstract— **It is well known that off-line programming (OLP) is an efficient control mode for industrial robots. However, OLP is not yet commonly and widely used in applications, since commercial OLP systems are very expensive, with prices even much higher than those of robotic systems themselves. Simple but effective OLP systems are desired, and such systems may be made based on other commonly used CAD systems. We have developed a robotic OLP system, which we called RobSim, based on SolidWorks and with Microsoft Visual Studio 2010. RobSim is developed as an add-on tool for the commonly used CAD software SolidWorks. With this OLP system, an object can be made, imported or modified in the conventional mode in SolidWorks environment, and various trajectories for a robot can be easily and conveniently defined and modified in the same environment. Trajectory interposition, kinematic computation, dynamic and graphic simulation can be conducted. Finally executable codes can be generated for the robot to perform tasks. In this paper, the development of RobSim is demonstrated. Specifically, the architecture of RobSim, the method for extracting position and orientation from a pre-defined path on an object for the robot tool, and path transformation, are presented. Simulation and experiments are also conducted to verify the effectiveness of the OLP system.**

I. INTRODUCTION

As well known, robots play important roles in modern industrial production. Motion planning and control of a robot is one of the basic problems to be solved for industrial applications such as welding, polishing, painting, carving and so on. Compared with teaching-and-playback method, off-Line Programming (OLP) is a more effective and efficient mode to plan and control motion of robots for many industrial applications, especially for those tasks on complex surfaces or along complicated trajectories, as aforementioned. A OLP system allows engineers to simulate production first in a visual graphic environment, and then control a real robot to implement the physical motion with the control system [1], [2], [3].

OLP systems mainly come from three sources, which are professional robotic software producers, robot manufacturers and research institutions. The OLP systems from the first two sources are usually commercial and very expensive. Therefore, many institutions developed their own systems based on CAD software or toolkits, such as AutoCAD, SolidWorks, OpenGL and VRML. Typical OLP softwares include Tecnomatix Robcad, DELMIA D5/V5, Robotworks and so on.

Tecnomatix Robcad enables design, simulation, optimization, analysis, off-line programming of multi-devices and manufacturing process for its production resources [4]. With Robcad, manufacturers are able to achieve desired manufacturing ideas to practical automatic manufacturing process. DELMIA is a widely used software system combining robotics and manufacturing for industrial applications [5]. In a 3D simulation environment, DELMIA can test and optimize the robot programs. The outstanding features of DELMIA are the "toolboxes", which are available to program with numerous functions. These functions are useful in various special areas, such as target definition, reachability analysis, collisions testing, path planning, etc. RobotWorks is a robotic CAD system running on SolidWorks, especially useful for workcell design, motion simulation and path programming [6]. These function modules use the SolidWorks graphics and internal engine to create a robot path quickly along faces, edges and curves of CAD objects.

Other OLP softwares developed by robot manufacturers include RobotStudio (ABB), KUKA.Sim Pro (KUKA) and MELFA Works (Mitsubishi) etc. RobotStudio provides various tools to increase efficiency of manufacturing tasks such as welding, polishing, teaching and multi-devices coordinate motion planning [7]. KUKA.Sim Pro is developed for off-line programming of KUKA robots, which is connected to the virtual KUKA controllers, equipped with cycle time analysis and the generation of robot codes [8]. It can upload CAD models from other CAD softwares.

MELFA-Works [9] is an add-on tool for SolidWorks products, which can automatically generate robot codes from existing CAD models with little installation and commissioning time. Developed by research institutions as educational robotics software, OLP systems are usually open source for academic purpose. References [10], [11] presented efficient robotic OLP methods based on DXF files of 3D modeling softwares AutoCAD. They decodes DXF files that describe the 3D position information of graphic elements such as points, lines, arcs and curves, then transforms into cartesian values. Similar approaches were also proposed with SolidWorks API SDK package [12], [3]. A robot simulation system based on OpenGL with VC++6.0 was presented in [13].

Although there are many robotic simulation and OLP

All authors but H. Li are with the School of Mechanical and Electrical Engineering,Guangdong University of Technology, Guangzhou, China, 510006. Y. Guan is the corresponding author, email: ysguan@gdut.edu.cn. H. Zhang is also with the Department of Computing Science, University of Alberta, Edmonton, AB, Canada, T6G 2E8. H. Li is with Guangdong Inspection and Quarantine Technology Center, China.

*The work in this paper is supported by the National Natural Science Foundation of China (Grant No. 51375095), the NSFC-Guangdong Joint Fund (Grant No. U1401240), the Natural Science Foundation of Guangdong Province (Grant No. S2013020012797, 2015A030308011), the State International Science and Technology Cooperation Special Items (Grant No. 2015DFA11700), the Frontier and Key Technology Innovation Special Funds of Guangdong Province (Grant No. 2014B090919002, 2015B010917003), and the State General Administration for Quality Supervision, Inspection and Quarantine (Grant No. 2014IK186).

978-1-4673-9675-2/15/$31.00 © 2015 IEEE

systems, most of them are not accessible. Those developed by professional software producers or robotic manufactures are usual of high prices, and those by institutions are not open to other organizations. And some systems lack of sufficiently excellent 3D modeling and path extracting functions. In this paper, we investigate how to develop an effective OLP system based on commonly and widely used software tools such as SolidWorks and Microsoft Visual Studio 2010. The system, which we called RobSim, includes mainly three function modules: which are path processing (extracting, optimizing and visualizing), motion simulator, and executable code generator. The SolidWorks API and RobSim architecture are introduced in Section II, and mathematic basis for three sorts of graphic path elements (line, arc and bezier curve) presented in Section III. Section IV describes path processing, including configuration extraction and transformation from pre-defied paths, and singular configuration avoidance with robotic kinematics. The simulation and experiment are conducted in Section V, and conclusions are drawn in the last section.

II. System Architecture

For friendly interaction with common users, excellent GUI and function modules layout of a robotic OLP system are required. Unfortunately, many of current OLP systems developed with OpenGL, based on AutoCAD or others, are not sufficiently convenient and even not capable to achieve these sound effects. OpenGL is good for circumstance rending and visual simulation of simple mechanical parts, but difficult for complex ones. AutoCAD is an excellent and professional CG modeling software tool for plane drawing, but weak in 3D modeling. As a OLP system, powerful 3D modelling and simulation are required, and significant features of a robot are included: a) robot configuration can be defined as a multi-rigid bodies mechanical system; b) different part are constrained by kinematic pairs; and c) joint motions are translational or rotational according to kinematic pairs. For these requirements, the CAD software tool SolidWorks is a good choice for developing a OLP system with its abundant Application Programming Interface (API).

A. SolidWorks API

As an outstanding CAD software system, SolidWorks has excellent performance in 3D modeling, visualization and simulation, with rich API containing hundreds of functions for Visual Basic 6.0, Visual Basic for Application (VBA), Microsoft Visual studio® 2010. The functions belong to some predefined object models and provide direct access to SolidWorks functions, such as creating a plane, extruding a body, inserting an existing part into a assembly document and verifying the parameters of a surface. Partial graphical hierarchy diagram of the interface inheritance is shown in Fig.1. SolidWorks® 2012 SP0 and Microsoft Visual studio® 2010 are chosen to develop our system. The open source API of SolidWorks is employed in RobSim.

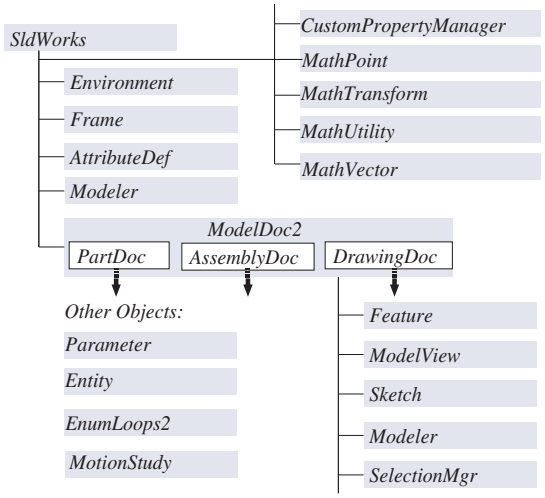


Fig. 1. SolidWorks API interface inheritance hierarchy diagram

B. RobSim Architecture

There are no existing add-on or toolbox in SolidWorks for robotic kinematic, dynamic, motion planing and so on. RobSim is built as a robotic add-on to SolidWorks, by adding graphics, kinematics engine and other modules. Solid object models can be directly accessed in SolidWorks graphics window, so that hidden links of objects are bridged between two software blocks, and objects motion is driven inside the SolidWorks assembly document. In this way, complex kinematics calculation and motion planning process can be added in RobSim. Fig.2 shows the architecture of RobSim, which clarifies interacted relationship between RobSim and SolidWorks.

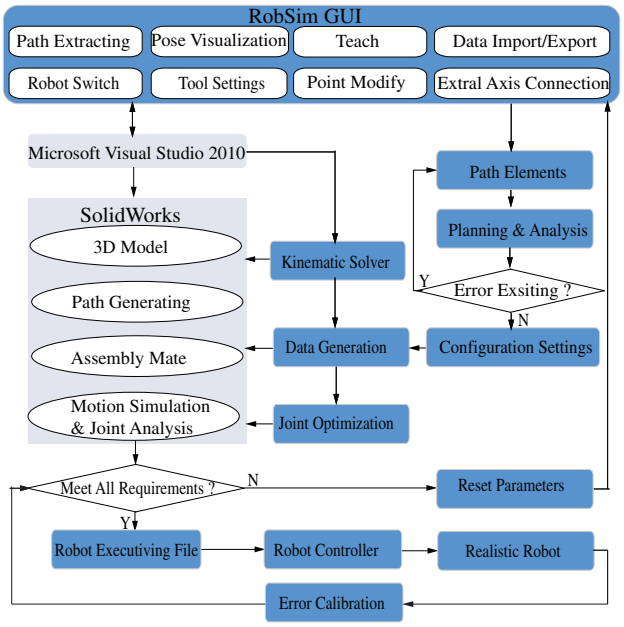


Fig. 2. Relationships between SolidWorks and RobSim

The following functional modules are shown in the architecture diagram.

- *3D Model Builder:* This module builds the 3D CAD models of parts, robots and their environments with SolidWorks. CAD models can also be made with other 3D CG tools, like Pro/E and UG. Many CAD models of commercial industrial robots can be found in websites. The CAD models are collected in a robotic library of RobSim. End-effectors or other accessories need to be designed or imported by users, and assembled with the robot models in the SolidWorks assembly document environment.
- *Path Generator:* Path generating includes path drawing and path extracting from entities. Line elements in different styles, like component edges, sketches, or spline curves, can be created with SolidWorks UI commands. As robot paths, the line elements will be planned into motion sequence by functions of RobSim. In this process, the graphic models always play important role between SolidWorks and RobSim.
- *Kinematic Solver:* This module defines the mapping between the joint space of a robot and the Cartesian space of the end-effector. Forward kinematics and inverse kinematics of robots are established by this module.
- *Configuration Visualizer:* After path calculation, configurations (positions and orientations) at discrete path points can be visualized with a series of short coordinate arrows. Visualization is achieved using the SolidWorks OpenGL graphical interface.
- *Motion Simulator:* There are two kinds of motion developed in RobSim, which are motion simulation of the end-effector and that of the manipulator. With the help of Motion Analysis in SolidWorks, the profile charts of the joint displacement, joint velocity, and joint acceleration can be rapidly exported. These profiles hence make motion simulation available and convenient.
- *Error Calibrater:* This module is to find out the errors between the simulated environment and the real world. This calibration processes relative accuracy and absolute accuracy. The workpiece frame and the tool frame with respect to the robot base frame need to be calibrated. Implementation algorithms are integrated in RobSim.
- *Robot Controller:* This module is to download executable codes generated by RobSim into the realistic robot controller. Reading the executable codes, the realistic robot conduct the desired motion.

III. Math Basis for Graphic Elements

To ensure that the robot end-effector can smoothly move along the paths extracted from the SolidWorks, transformation and interposition must be executed [14], [15]. In this section, robotic rules for three graphic elements (line, arc and Bezier curve) are introduced. According to the rules, mathematical models of these graphic elements are established.

A. Line

A line segment can be defined by two points (the start point and the end point). In order to discretize the line, we can commonly parameterize the linear equation with a point (x_0, y_0, z_0), a direction vector $\overrightarrow{d} = (m, n, p)$ and the parameter t_i. Therefore, An arbitrary point $p_i(x_i, y_i, z_i)$ on the line can be described by the following equations

$$x_i = x_0 + t_i m, \quad x_i = y_0 + t_i n, \quad z_i = z_0 + t_i p \tag{1}$$

Discrete interposition points are thus simply obtained with the above equations. However, orientation information for a regular robotic path is needed. Therefore the orientation information along the line is to be extracted. According to geometry, spatial edge of a body lies on two adjacent faces A_1 and A_2, that is, the intersection of them. The normal vectors $\overrightarrow{n_1} = (u_1, v_1, w_1)$ and $\overrightarrow{n_2} = (u_2, v_2, w_2)$ of the two surfaces are obtained, respectively. Then, an appropriate coordinate system with the line direction vector and one of the surface normal vectors can be built up, as shown in Fig.3.

With one of the surface normal vectors $\overrightarrow{n_1}$, the configuration T_i at the i-th point p_i on the line can be defined by the following homogeneous matrix.

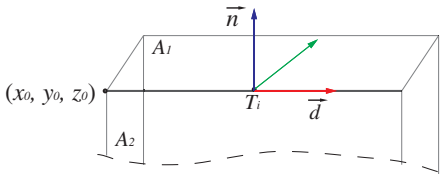


Fig. 3. Spatial line expression

$$T_i = \begin{bmatrix} \overrightarrow{d} & \overrightarrow{n_1} \times \overrightarrow{d} & \overrightarrow{n_1} & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} m & pv_1 - nw_1 & u_1 & x_i \\ n & mw_1 - pu_1 & v_1 & y_i \\ p & nu_1 - mv_1 & w_1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B. Arc

Arc edge of 3D model in SolidWorks can be extracted in a similar manner. The start point $s(s_x, s_y, s_z)$, the end point $e(e_x, e_y, e_z)$, the circular center $o(o_x, o_y, o_z)$, the normal vector $\overrightarrow{n} = (u, v, w)$ and the radius r are accessible from the SolidWorks API function, as shown in Fig.4. Then, if incremental value Δ_i is set, any point $p_i(x_i, y_i, z_i)$ on the arc can be calculated by the following equations

$$\begin{aligned} v_1 &= s_x - o_x + \Delta_i * (e_x - s_x) \\ v_2 &= s_y - o_y + \Delta_i * (e_y - s_y) \\ v_3 &= s_z - o_z + \Delta_i * (e_z - s_z) \\ kt &= \frac{R}{\sqrt{v_1^2 + v_2^2 + v_3^2}} \\ x_i &= o_x + kt * v_1 \\ y_i &= o_y + kt * v_2 \\ z_i &= o_z + kt * v_3 \end{aligned} \tag{2}$$

where v_1, v_2, v_3 and kt are intermediate variables.

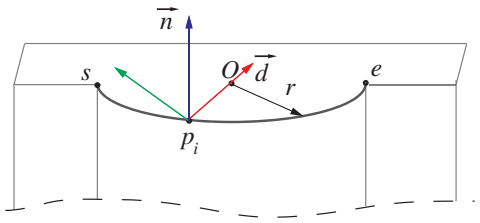


Fig. 4. Spatial arc expression

For the orientation, the coordinate system can be obtained according to the geometrical relationship between the point p_i and the circular center O (refer to Fig.4), which is defined as $\overrightarrow{d} = (m, n, p)$. Hence, the configuration T_i at the i-th point p_i on the arc can be defined by the following homogeneous matrix.

$$T_i = \begin{bmatrix} \overrightarrow{d} & \overrightarrow{n} \times \overrightarrow{d} & \overrightarrow{n} & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} m & pv - nw & u & x_i \\ n & mw - pu & v & y_i \\ p & nu - mv & w & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

C. Bezier Curve

Bezier curve is defined by the vertices of a polygon that encloses the resulting curve [16]. The effects of the vertices are weighted by the corresponding blending functions and blended as in the Hermite curve. We can derive the controlling vertices c_j of the controllable polygon by the SolidWorks API functions. The equation of the Bezier curve can be obtained as

$$p(u) = \sum_{i=0}^{n} \begin{pmatrix} n \\ i \end{pmatrix} u^i (1-u)^{n-i} p_i \qquad 0 \le u \le 1 \qquad (3)$$

where

$$\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}$$

Now we can verify that the Bezier curve defined by the above equation satisfies our target to tessellate the curve by parameter u. Through the tessellation points, we can obtain the interposition point $p_i(x_i, y_i, z_i)$ of Bezier curve.

For the orientation, we need to derive the expression of the Bezier curve derivatives. This expression described the relation between the derivatives of Bezier curve and their original control points. We then differentiate the equation with respect to u:

$$\begin{aligned} \frac{dp(u)}{du} =& \sum_{i=0}^{n} i \begin{pmatrix} n \\ i \end{pmatrix} u^{i-1}(1-u)^{n-i} p_i - \\ & \sum_{i=0}^{n} (n-i) \begin{pmatrix} n \\ i \end{pmatrix} u^i (1-u)^{n-i-1} p_i \\ =& n \sum_{i=0}^{n} \begin{pmatrix} n-1 \\ i \end{pmatrix} u^i (1-u)^{n-i-1} p_i (p_{i+1} - p_i) \end{aligned} \qquad (4)$$

The tangent vector $\overrightarrow{tan} = (t_x, t_y, t_z)$ at an arbitrary tessellation point can be obtained. To set up a coordinate system, another vector containing the corresponding point must be found. For this end, we define a vector $\overrightarrow{n} = (n_x, n_y, n_z)$ pointing from the current tessellation point $p_i(x_i, y_i, z_i)$ to the next one $p_{i+1}(x_{i+1}, y_{i+1}, z_{i+1})$. Since the vectors $\overrightarrow{tan}$ and $\overrightarrow{n}$ intersect at point p_i, the cross product result is

$$\overrightarrow{a} = \overrightarrow{n} \times \overrightarrow{t}$$

From above discussion, a Bezier curve is specifically shown in Fig.5. With the position and orientation at a tessellation point p_i of Bezier curve, we can derive the following homogeneous matrix

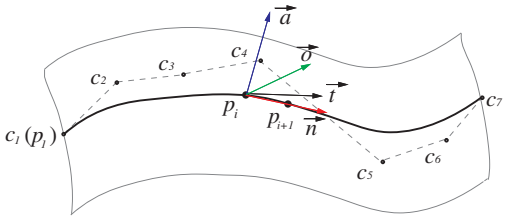


Fig. 5. Spatial Bezier curve expression

$$T_i = \begin{bmatrix} \overrightarrow{n} & \overrightarrow{n} \times \overrightarrow{a} & \overrightarrow{a} & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x & n_y a_z - n_z a_y & a_x & x_i \\ n_y & n_z a_x - n_x a_z & a_y & y_i \\ n_z & n_x a_y - n_y a_x & a_z & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

IV. Path Transformation

For configuration transformation, Denavit-Hartenberg (D-H) method [17] uses the four parameters (a, α, d, θ) associated with a particular convention for attaching reference frames to the links of a spatial kinematic chain. The matrix of configuration transformation from coordinate frame n to coordinate frame $n-1$ is given by the following equation.

$${}^{n-1}_{n}T = \begin{bmatrix} c\theta_n & -s\theta_n & 0 & a_{n-1} \\ s\theta_n c\alpha_{n-1} & c\theta_n c\alpha_{n-1} & -s\alpha_{n-1} & -d_n s\alpha_{n-1} \\ s\theta_n s\alpha_{n-1} & c\theta_n s\alpha_{n-1} & c\alpha_{n-1} & d_n c\alpha_{n-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

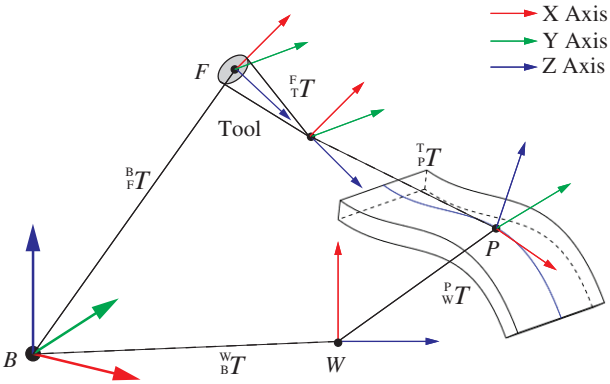


Fig. 6. Coordination transformation

where $c\theta_n$ is a short note for $\cos\theta_n$, and $s\theta_n$ for $\sin\theta_n$.

During path planning, path transformation and workpiece calibration are in need. Fig.6 shows the relationship between the frames at a discrete path point, the workpiece and the robot base, where five transformation matrixes, ${}^B_F T$, ${}^F_T T$, ${}^T_P T$, ${}^P_W T$, ${}^W_B T$, are to be calculated. ${}^B_F T$ represents the transformation from the robotic wrist frame to the base frame, which can be gained by D-H method according to the link and joint parameters. ${}^F_T T$ is calculated after a Tool Center Point (TCP) frame is defined on the tool. ${}^T_P T$ is the transformation from the path point frame on the workpiece to the robot end-effector frame. ${}^P_W T$ describes the transformation from the world frame to the the path point frame. And ${}^W_B T$ describes the relationship between the robot base frame and the world frame. The following equation can be obtained.

$$ {}^B_F T \; {}^F_T T \; {}^T_P T = ({}^P_W T \; {}^W_B T)^{-1} \tag{5} $$

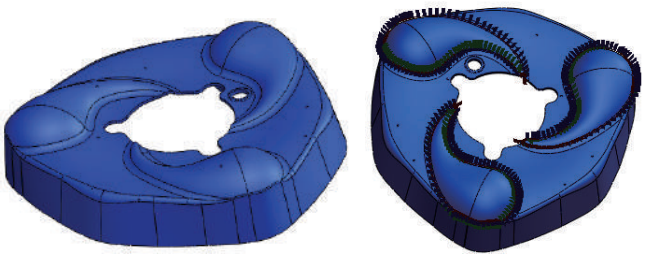
Finally, we can calculate the inverse kinematics from Cartesian space ($\Re$) to joint space(Θ) for path following,

$$ \Theta = IK(T) \tag{6} $$

V. Simulation and Experiment

To verify the effectiveness of the presented method above and demonstrate the functions of OLP system RobSim, simulation and experiment are conducted in this section. The task is to control the robot end-effector to follow curves on the cover of our suction module as shown in Fig.7(a), to mimic application of welding along complex paths. The paths are specified directly on the CAD model of the workpiece (the suction module cover), and the poses of the robotic tools are then defined along the paths and visualized, as shown in Fig.7(b). The key steps of the software implementation are presented in Fig.8.

Motion of the robot is simulated first with the RobSim system. Once the simulation is successful, executable codes are generated and downloaded into the robot controller for real implementation. Fig.9 (a)-(d) are some snapshots of the simulation in the SolidWorks and the experiment with a real robot. Fig.10 is shown the main several modules of RobSim in the SolidWorks environment. It takes less than thirty minutes to finish the whole experiment, including 3D model importing, path defining and extracting, tool installation, simulation, downloading executable codes to robot controller and running. The performance of the OLP system is satisfied.



(a) CAD model of the workpiece (b) Tool pose along the paths

Fig. 7. Workpiece model and pose visualization

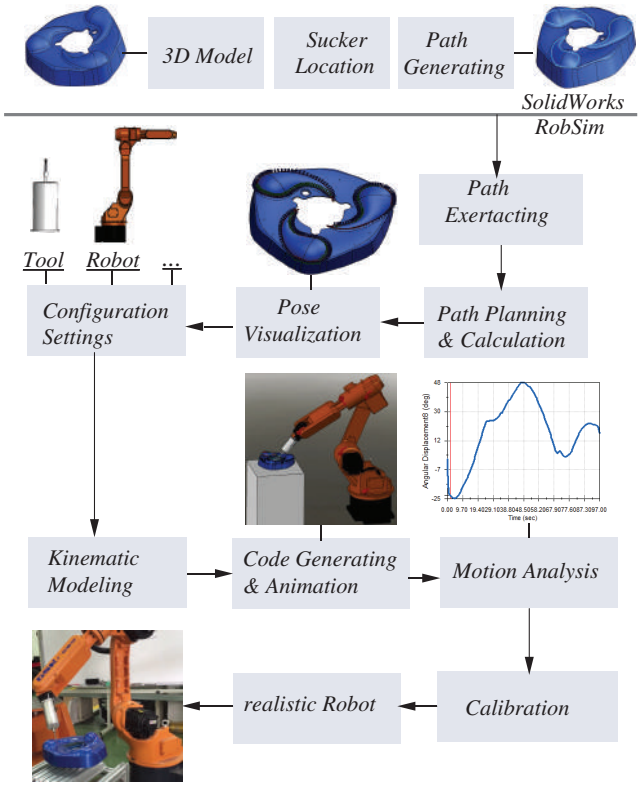


Fig. 8. Key steps of software implementation

VI. Conclusions and Future Work

As presented in this paper, a robotic off-line programming (OLP) system, RobSim, has been developed. Based on SolidWorks, the presented OLP system has been built with Visual Studio 2010, and can be used as an added-on tool in SolidWorks. Like other commercial OLP systems, it has basic OLP functions such as 3D object modeling, definition and modification of complex trajectories for manipulators, position and orientation extraction and interpolation, simulation, executable code generation. Simulation and experiment have verified the effectiveness of the built simple OLP

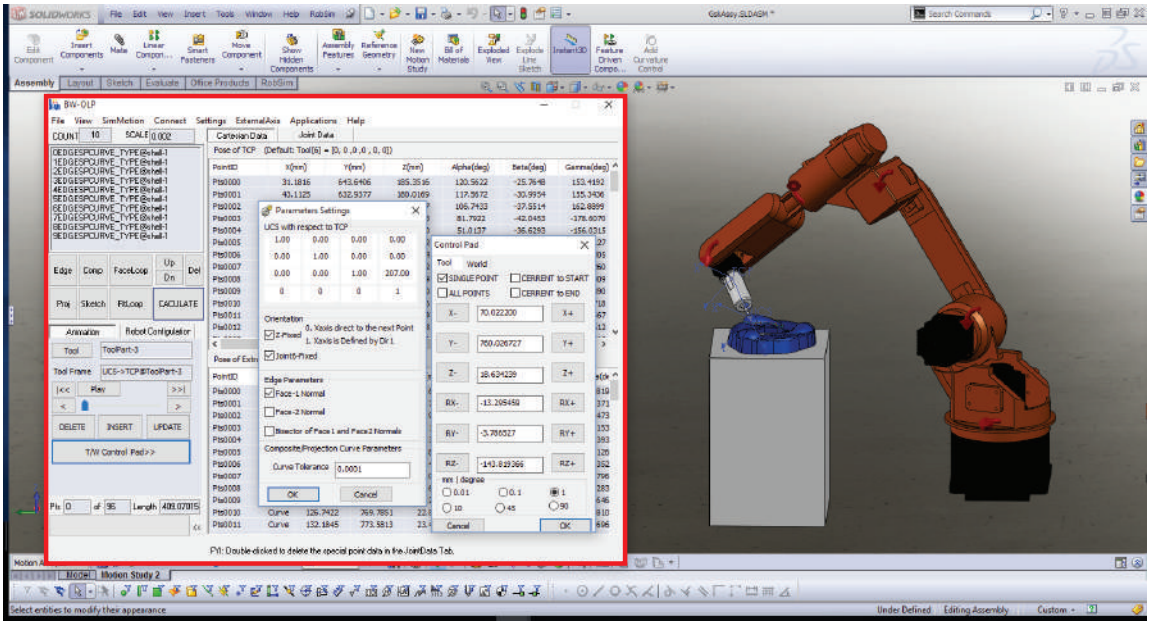
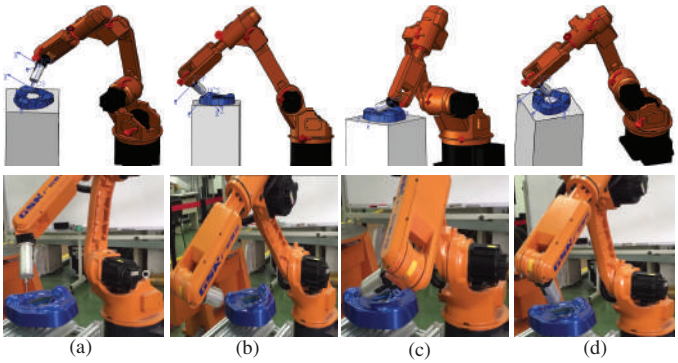


Fig. 10. Global view of RobSim in SolidWorks software



(a) (b) (c) (d)

Fig. 9. Simulation and Experiment snapshots

system and illustrated its potential application of welding. Other potential applications such as painting, deburring, polishing, and carving may be also implemented with this OLP system. We believe that the architecture of RobSim, the presented method for extracting and interpolating positions and orientations from the pre-defined trajectories are helpful to develop similar OLP systems using other commonly used CAD softwares.

However, RobSim is not yet a complete LOP system. It still lacks of many important functions or features such as collision detection in simulation and on-line calibration. In the near future, these functions will be implemented and added to RobSim. Besides, the system functions will also be extended to integrate coordination of multiple robots.

REFERENCES

[1] Y. Yong, J. Gleave, J. Green, and M. Bonney, *Off-line programming of robots*. Wiley, 1985.

[2] E. Freund, D. Rokossa, and J. Roßmann, "Process-oriented approach to an efficient off-line programming of industrial robots," in *IECON'98, Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 208–213, 1998.

[3] Y. Gan, X. Dai, and D. Li, "Off-line programming techniques for multirobot cooperation system," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[4] S. Wang, J. Zhang, H. Cai, Y. Wan, and Y. Fu, "An advanced robot simulation system-robcad," *Journal of Harbin Institute of Technology (in Chinese)*, vol. 3, p. 021, 1993.

[5] Z. Bzymek, M. Nunez, M. Li, and S. Powers, "Simulation of a machining sequence using delmia/quest software," *Computer-Aided Design and Applications*, vol. 5, no. 1-4, pp. 401–411, 2008.

[6] A. R. MFF, W. Mansor, W. Muhamad, S. Mohd Zaidi, *et al.*, "Design and simulation of robotic spot welding system for automotive manufacturing application," 2005.

[7] "Abb robotics: Operating manual robotstudio," *Västerås, Sweden*, 2007.

[8] K. Vollmann, "A new approach to robot simulation tools with parametric components," in *ICIT'02, IEEE International Conference on Industrial Technology*, vol. 2, pp. 881–885, 2002.

[9] V. Bottazzi and J. Fonseca, "Off-line programming industrial robots based in the information extracted from neutral files generated by the commercial cad tools," *Industrial Robotics: Programming, Simulation and Applications*, p. 349, 2006.

[10] Z. Yin, Y. Guan, *et al.*, "Off-line programming of robotic system based on dxf files of 3d models," in *ICIA'13, IEEE International Conference on Information and Automation*, pp. 1296–1301, 2013.

[11] P. Neto, J. N. Pires, *et al.*, "Robot path simulation: a low cost solution based on cad," in *RAM'10, IEEE Conference on Robotics Automation and Mechatronics*, pp. 333–338, IEEE, 2010.

[12] S. Mitsi, K.-D. Bouzakis, G. MansouG, D. Sagris, and G. Maliaris, "Off-line programming of an industrial robot for manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 3, pp. 262–267, 2005.

[13] W. Zhang, F. Zou, D. Qu, and F. Xu, "Research of key technologies on 3d simulation system of industrial robot," in *The 7th World Congress on Intelligent Control and Automation*, pp. 565–568, 2008.

[14] K. Hirasawa, Y. Shimizu, and T. Matsumura, "Time-based interpolation control of a robot," July 28 1987. US Patent 4,683,543.

[15] H. Koyama, T. Asano, F. Noguchi, and S. Fujinaga, "Robot interpolation control method," Aug. 25 1987. US Patent 4,689,756.

[16] K. Lee, *Principles of CAD/CAM/CAE systems*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[17] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.