RESEARCH ONLINE

University of Wollongong
Research Online

Faculty of Engineering - Papers (Archive)

Faculty of Engineering and Information Sciences

2012

Automated offline programming for robotic welding system with high degree of freedoms

Zengxi Pan
University of Wollongong, zengxi@uow.edu.au

Joseph Polden
University of Wollongong, jpolden@uow.edu.au

Nathan P. Larkin
University of Wollongong, nlarkin@uow.edu.au

Stephen van Duin
University of Wollongong, svanduin@uow.edu.au

John Norrish
University of Wollongong, johnn@uow.edu.au

http://ro.uow.edu.au/engpapers/5042

Publication Details

Pan, Z., Polden, J., Larkin, N. P., van Duin, S. & Norrish, J. (2012). Automated offline programming for robotic welding system with high degree of freedoms. Lecture Notes in Electrical Engineering, 121 685-692.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

RESEARCH ONLINE

Automated Offline Programming for Robotic Welding System with High Degree of Freedoms

Zengxi Pan, Joseph Polden, Nathan Larkin, Stephen van Duin, and John Norrish

Faculty of Engineering, University of Wollongong, NSW, 2522, Australia
{zengxi,jwp973,nlarkin,svanduin,johnn}@uow.edu.au

Abstract. Although robotics based flexible automation is an intriguing prospect for small to median enterprises in the era of the global competition, the complexity of programming remains one of the major hurdles limiting its applications. This paper presents an automated offline programming (AOLP) method to address this issue. AOLP is software that automatically plans and programs for a robotic welding system with high Degree of Freedoms (DOFs). It takes CAD model as input, and is able to generate the complete robotic welding code without any further programming effort.

Keywords: offline programming, lean automation, welding, CAD, high DOFs.

1 Introduction

Although robotics based flexible automation is an intriguing prospect for small to median enterprises (SMEs) in the era of the global competition, the complexity of programming a robotic system remains one of the major implementation challenges. In an industrial environment, there are two main methods of robot programming; online programming (including lead-through and walk-through) and offline programming (OLP) [1]. Manual online programming requires no additional hardware and software other than those to be used for the manufacturing process. However, the generated program is very inflexible and it can only handle simple robot paths. On the other hand, while OLP methods can generate flexible robot programs for complex robot paths, its high cost can only be justified for a large production volume. Also, programming a robotic system using typical commercially available OLP software is still a manual process. It does not reduce the programming overhead. Instead, OLP shifts the burden of robot programming from the robot operators jogging robot manipulator in the workshop to the software engineers, who 'jog' a simulated robot in a computer modeled environment. OLP provides advantages such as reusable code, flexibility for modification, and less system downtime during programming stage. However, the time and cost required to generate code for a complex robotic system using OLP is expected to be similar to if not more than that using online programming. Currently, for a complex manufacturing process with small to median production volume, very few robotic automation solutions are used to

Y. Wu (Ed.): Advances in Computer, Communication, Control & Automation, LNEE 121,, pp. 685–692.
springerlink.com © Springer-Verlag Berlin Heidelberg 2011

replace manual production due to this expensive and time-consuming programming overhead.

An example is in armored vehicle production where over 500 welds are required to assemble a large vehicle hull from steel plates. For this complex workcell involving robots with 13 Degrees of Freedom (DOFs), manual online program methods required more than six months to generate the program, while the cycle time of the welding process itself is only sixteen hours. In this case, the programming time is approximately 90 times of the production time. When the vehicle has a few variants and the production volume for each variant is low, the manufacturer would rather go back to manual welding process even though the robotic welding system was setup, simply due to the long programming time. A more efficient and cost-effective robot programming method needs to be developed to address this issue.

This paper presents an automated offline programming (AOLP) solution to automatically plan and program a robotic welding system with high DOF manipulators. It uses a CAD model as input, and is able to generate the complete robotic welding code without any further programming effort. AOLP can generate collision-free and singularity-free trajectories for the complete system including the linear rail, auxiliary positioning robot, and welding robot automatically. Following this introduction Section, Section two describes the configuration of the welding system. The components and steps of AOLP are presented in Section three and Section four respectively. Section five presents the performance of AOLP. A summary and some discussions are provided in Section six followed by acknowledgement and references.

2 The Robotic System

The layout and CAD model of the robotic welding workcell are shown in Fig. 1. Due to the high number of seams to be welded and the complex hull geometry, a specific robotic cell was designed to maximize the number of external and internal seams that can be reached by the welding robot. To satisfy the cycle time requirement, the final design of the cell includes two identical welding systems and two preheating systems. The welding system is a robot-on-robot-on-rail setup, while the preheating system is a robot-on-rail setup. There are a total of six articulated robots and four linear rails in the workcell.

Each welding system in the cell consists of a small welding robot, a large auxiliary robot and a linear rail. The small welding robot is mounted on the wrist of the large auxiliary robot, which is sitting on the linear rail. The vehicle hull is mounted on a rotating trunion to allow the welding robot to maintain a down-hand welding position and provide access to areas such as the roof of the hull, or internal access through an opening such as the windscreen frame. The robotic cell features sensors such as a laser profile scanner and pyrometer to aid in the re-calibration and accurate welding of each individual seam. The complexity of this robotic system poses many difficulties for programming either using online or offline methods.

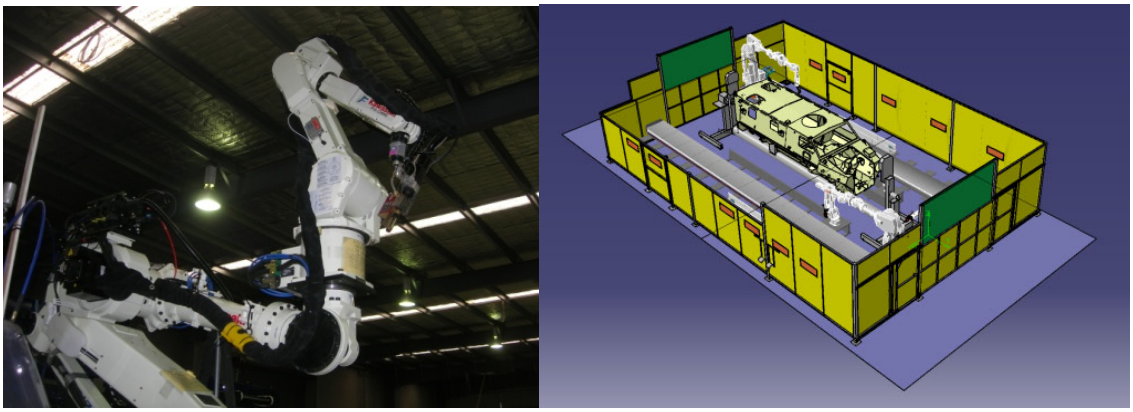

Fig. 1. (a) Robotic system for vehicle hull welding developed by RTA Automation (b) CAD model of robotic system

3 The Elements of AOLP

3.1 Robot Kinematics

The DH (Denavit-Hartenberg convention) models of the industrial manipulators used in the system were constructed. For the convenience of calculation, the robotics toolbox for MATLAB [4] was used for forward kinematics calculation. The inverse kinematics was formulated using closed-form symbolic equations based on [5], which is much more efficient than iterative solutions. Also, unlike the iterative method, which can only find a solution closest to the given reference joint target, the closed form method is able to find solutions for all possible robot configurations, which is much more useful in robot trajectory and motion planning. Currently the inverse kinematics of Kawasaki FS06L, ZX300, ABB IRB4400 [6, 7] robots have been formulated. Note that this close form inverse kinematic solution is only suitable for robots with 6 DOF that feature a spherical wrist.

3.2 CAD Representation and Clash Model

As the clash detection among the different parts in the system, such as robot, tool and environment, is the most computationally intensive, a precise and prompt collision detection method is critical for successful OLP software. A simplified bounding volume method is used here for efficient collision checking, where speed is more important than exactness and visualizations [8]. The clash model for the moving parts (robot, welding torch and other tools attached) and the fixing parts (the workpiece, trunion, other fixtures, etc) are modeled differently for fast clash check, as shown in Fig. 2.

A sphere-bounding model was used to model the robot and tool. A group of spheres with different diameters represent the robot and tool as collision detection using spheres is most efficient. Use of a multi-level sphere model was implemented with fewer spheres providing a rough collision check and more spheres for final verification. The clash model for the work piece, fixtures and environment has been modeled using a modified bounding box model (‘plate’ model), which treats all objects as a plate with vertexes on a surface, a normal direction and thickness.

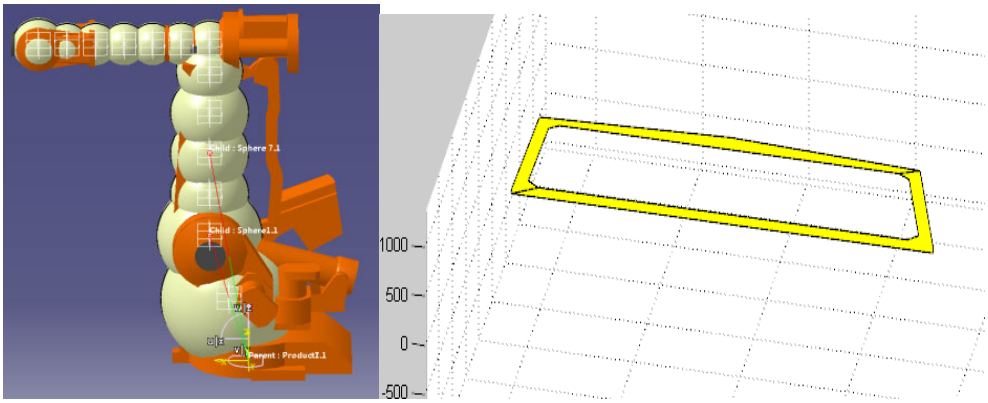

Fig. 2. (a) Sphere model of an ABB IRB4400 robot; (b) 'Plate' model of the workpiece

Since the objects in the workcell are modeled using very simple geometries, the collision check becomes very efficient. There are only two different cases, collision between a sphere and a plate (robot-workpiece, tool-workpiece) or between two spheres (robot-tool). Both in-process motions and transition motions compute distance check along the path, and search if necessary to avoid collision and maintain a reasonable distance between tool/robot and obstacles.

3.3 Path Planning

The search algorithm for path planning must produce weld motions that are smooth, collision free, and within the reach of the robot, avoiding singularities and joint limits. Additionally, the search must produce paths that are as process-optimal as possible in terms of torch pose relative to weld geometry. It is further desired that the planning algorithm be very fast when the planning problem is simple, but able to find solutions for difficult problems when they arise [2].

The complexity of path planning grows exponentially with the number of DOFs, which is well known as curse of dimensionality. Since the whole system has 13 DOFs with a robot-on-robot-on-rail setup, a brute-force search through all robot configurations is very time consuming and not practical if the system is treated as a single mechanism. As the large auxiliary robot is used to improve the reachability of the system and is stationary during welding, path planning is broken into three steps:

1) Auxiliary rail and robot positioning,
2) Weld torch placement, and
3) Weld path planning.

While searching a path for the welding robot, the motion of auxiliary robot/rail is sampled to provide base position for the welding robot, which is considered as a tool attached on the auxiliary robot with a certain pose. This method dramatically reduces the complexity of the AOLP process, avoiding the case of selecting an optimal solution from an infinitive number of solutions.

Transition motions between the welding paths are computed using Probabilistic Roadmap (PRM) algorithm [9,10,11]. The algorithm computes random trajectories in joint space and attempts to connect the trajectories to seek a feasible path. The PRM

has shown to be very successfully in solving difficult path planning problems in high dimensional configuration spaces.

4 The Steps of AOLP

4.1 Preparation

To make the software easy to use for robot operators with limited computer skills, an Excel spreadsheet is created at the preparation stage as the input for the programming stage in AOLP. Information including sphere-bounding models for the robots and tools, 'plate' model for the workpiece and fixtures, tag definition of welding seams, zone information for auxiliary robot/rail to position the welding robot, are saved in a single Excel spreadsheet.

The spreadsheet is generated using a manual or automated processes in DELMIA, a commercial OLP package. Using a combination of VB script and manual selection, the CAD information for robots, tooling, workpieces, and fixturing is converted into the MATLAB sphere-bounding model or 'plate' model. To generate the 'plate' model of workpiece, the vertex coordinates of one surface on each plate are extracted from CAD in anti-clockwise order and the plate thickness calculated by selecting a vertex on the other surface. This data is later read by MATLAB to generate surfaces that can be used for collision detection in path planning.

Weld seams are also generated using DELMIA VB script. Each weld seam is defined by two tags, which denote the beginning and end of the seam along with nominal torch pose. When defining the tags, the user first selects the edge that represents the weld seam location followed by the two surfaces that form the seam to calculate the orientation of the torch at the welding position. This data is also stored in the Excel spreadsheet.

4.2 Robot Program Generation

As shown in Fig. 3, the user interface of the OLP has three panels from left to right; setup, OLP and Full path. In the Setup panel, the user selects which welding seam to program and AOLP will load workcell information such as trunion orientation from the Excel spreadsheet.

The first step is to determine possible positions for the auxiliary robot. Sampling the space around the appropriate hull opening for internal seams or space around the weld seam for external welds for collision and reachability establishes possible weld robot base positions. Typically 343 positions are tested with around 50 found to be suitable.

In Gas Metal Arc Welding (GMAW), there is an optimal torch orientation relative to the plates to be joined, and although the torch can be rotated around the axis of the filler wire, tool orientation is limited as to not affect weld quality. However, there is added freedom to reach welds into corners where optimal torch orientation is sacrificed for access. This adds complexity to the AOLP system as it needs to maximize flexibility to access difficult areas, but apply optimal the torch angle to maximize weld quality.

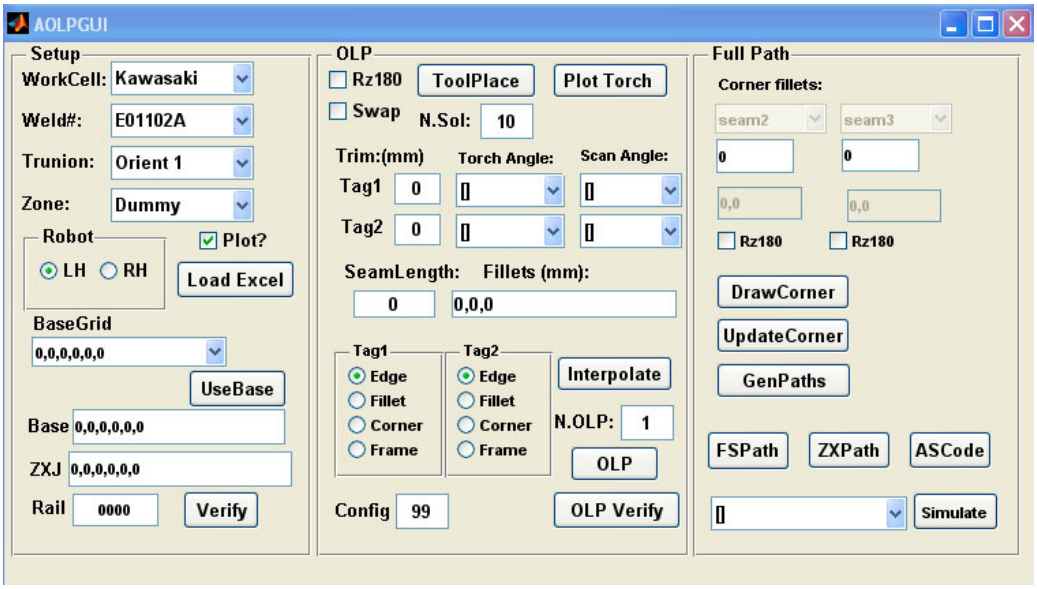

Fig. 3. User interface of AOLP

The torch placement algorithm samples the torch orientation space to determine a list of possible collision free torch paths without the robotic manipulator. The user can then manually pick the preferred angle or let the torch placement algorithm determine the most optimal solution.

After a good torch orientation is found, the next step is to find which auxiliary robot position to use to position the base of the welding robot and which robot configuration guarantees a collision free reachable solution for the welding path. The welding path is sampled in increments, currently 100mm, and each collision free auxiliary robot position is tested until a good solution is found.

A temperature measurement position is automatically selected in the middle of the weld seam. The temperature is measured before welding to ensure that the preheat requirement of the weld procedure is satisfied. Weld seams are also locally calibrated using a laser profile sensor. There are various options to calibrate the weld seam position depending on the seam geometry. For example, a corner position is determined by scanning the seam to be welded along with another seam that intersects with the weld corner. The user can select which calibration method to use for choose which other geometry is to be used. The AOLP algorithm incorporates the temperature measurement and calibration points into the generated collision free path.

With the weld path, including calibration and temperature measurement determined, the next step is to find the transition motions to translate the welding robot from the folded position to the start of the weld path and back to the folded position. A probabilistic roadmap based algorithm is used here to generate random assisted tags to build up the path.

The AOLP software also provides simulation features. The user can either display the simulation during the programming or after a robot code is generated. This provides useful information for the user to understand what is happening during the offline programming and help adjust parameters if the AOLP has troubles programming a specific seam.

A post-processor is an important element of an OLP system. Once the complete motion of the system is generated, it needs to be converted to the robot language of a specific robot manufacturer as well as adding all communication and I/O code [3]. The post-processor is able to translate output statements to the ABB RAPID and Kawasaki AS language.

5 Results

In flexible automation, reuse of a robotic system to manufacture variants of a single product or many different products is important and minimizing system downtime due to system reprogramming is a key technology to reducing cost and increasing flexibility. Due to the complexity of the aforementioned robotic system, it took more than six months to manually (online) program welds for a specific vehicle model. For any new model, reprogramming using similar conventional methods will result in similar lengthy programming times. It is estimated that each new seam will take average 4 hours to reprogram. For a new model with 200 new seams, this equates to a production loss of around 3 months simply wait for the system to be reprogrammed. Using off-the-shelf OLP software, the total programming time will be similar to online programming. However, as programming can take place offline, system downtime is minimized taking around 1 hour to test each new weld seam. Using AOLP, robot code for each seam takes less than 5 minutes to generate. The preparation stage, during which the Excel spreadsheet was set up with system model information, took 4 weeks. Similar system downtime to OLP is needed to test the generated code. The system programming time using each method is compared in Table 1.

Table 1. Comparison of programming time using online programming, OLP and AOLP (Unit: Weeks)

	Offline time (system setup and preparation)	System downtime	Total lead time
Online	2	12	14
OLP	8	4	12
AOLP	4	4	8

6 Summary

AOLP provides an automatic means of generating robot code, much faster and more reliably than previous attempts using either online or offline programming. By reducing the expense associated with programming overhead, it improves the commercial viability for robotics-based lean automation for manufacturing low volume or one-off products, which is common for small to medium enterprise (SMEs).

AOLP is independent of the process to be accomplished, or any particular robot or process hardware. This provides opportunity for a large domain of potential

applications in a variety of industries. Although the current implementation of AOLP is for welding applications, the process modeling and path planning capability has applicability to a number of other industrial applications.

Acknowledgments. The authors acknowledge the support of the Defence Materials Technology Centre (DMTC), which was established and is supported by the Australian Government's Defence Future Capability Technology Centre (DFCTC) initiative. In addition thanks are due to DMTC partners Thales Australia and RTA Automation.

References

1. Pan, Z., Polden, J., Larkin, N., Van Duin, S., Norrish, J.: Recent Progress on Programming Methods for Industrial Robots. In: ISR/ROBOTIK, Munich, Germany, (June 7-9, 2010)
2. Ames, A.L., Hinman-sweeney, E.M.: Automated Generation of Weld Path Trajectories (2005)
3. Chan, S.: Post-processing methodologies for off-line robot programming within computer integrated manufacture. Journal of Materials Processing Technology 139, 8–14 (2003)
4. Corke, P.: Robotics toolbox for Matlab, `http://petercorke.com/Robotics_Toolbox.html`
5. Paul, R., Zhang, H.: Computationally Efficient Kinematics for Manipulators with Spherical Wrists Based on the Homogeneous Transformation Representation. International Journal of Robotics Research 5(2), 32 (1986)
6. AS Language Refernce Manual, Kawasaki Industries, Ltd. (2002)
7. ABB IRB4400 product online manual, ABB Flexible Automation, M98
8. Jimenez, P., Thomas, F., Torras, C.: Collision detection algorithms for motion planning. In: J.-P. Laumond (ed.) Robot Motion Planning and Control, ch. 6 (1998)
9. Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996), doi:10.1109/70.508439
10. Geraerts, R., Overmars, M.H.: A comparative study of probabilistic roadmap planners. In: Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR 2002), pp. 43–57 (2002)
11. Bertram, D., Kuffner, J., Dillmann, R., Asfour, T.: An integrated approach to inverse kinematics and path planning for redundant manipulators. In: Proceedings 2006 IEEE International Conference on Robotics and Automation ICRA 2006, pp. 1874–1879 (2006)
12. RinasWeld, Kranendonk, `http://www.rinas.dk/RW__.htm`