Automated Programming for Robotic Welding

Nathan Larkin[1,2], Andrew Short[1], Zengxi Pan[1,2(✉)], and Stephen van Duin[1,2]

[1] School of Mechanical, Materials, Mechatronic and Biomedical Engineering, University of Wollongong, Wollongong, NSW 2522, Australia
zengxi@uow.edu.au

[2] Defence Materials Technology Centre, Hawthorn, Australia

Abstract. Robotic welding automation allows manufacturers to increase quality, flexibility and reduce costs. However, the costs involved in programming welding robots for small production runs limits viability for Small and Medium Enterprises to employ arc welding automation. This paper outlines an Automated Offline Programming framework which can be used to generate robot programs directly from Computer Aided Design models with minimal human input, allowing programming costs to be drastically reduced or even eliminated. The key stages of our approach are presented and a specific implementation for welding of complex pipe structures is shown. The results demonstrate the feasibility of our method to enable truly flexible robotic welding automation.

Keywords: Robotics · Welding · Automated programming · Motion planning

1 Introduction

Industrial robotic automation is typically the domain of high volume manufacturers with short cycle times and large batch sizes. The cost of tooling and programming limits the viability of robotic welding automation for low volume and once-off production. This paper presents an Automated Offline Programming (AOLP) approach for generating robotic welding programs directly from Computer Aided Design (CAD) models, reducing or eliminating programming costs.

Arc welding is a key robot application. 13% of robots sold in 2011 were used in arc welding applications [1]. Tooling costs for robotic arc welding are similar to those in manual arc welding, hence the programming costs involved is the key limiting factor for use in low production volumes. By reducing programming time and costs, the presented AOLP framework makes automated robotic welding feasible for Small and Medium Enterprises (SMEs).

There are two primary classifications of robot programming methods: automatic and manual programming [2]; and online and offline programming [3]. Conventionally robots use online manual programming (or lead-through programming), in which an operator uses a teach pendant to guide the robot through the designed welding motions.

These motions are recorded to create the robot program. This approach is simple and low cost, but requires: significant time, a physical robot, and is limited by the skill of the operator. Recent extensions to this focus on programming by demonstration, in which an operator demonstrates a task to perform which the robot then "learns" using sensors [4].

In contrast, Offline Programming (OLP) uses a model of the robot, cell and work piece to allow a user to generate robot programs using a software package without requiring exclusive access to a physical robot. By taking advantage of software tools and existing CAD data it is possible to rapidly develop and simulate robot programs. However, this programming process has many of the same challenges as conventional programming: it requires a skilled operator, there are challenges translating to a physical robot, and for some cases the costs can be higher than conventional programming [5].

Automated Offline Programming (AOLP) [5] is an approach which extends OLP by using algorithms to automate much of the robot programming process. This can drastically reduce or even eliminate human effort required for robot programming, making robot programming viable for low volume applications. It also allows generating complex motions to access challenging areas, as well as integrating sensing to deal with part placement inaccuracies.

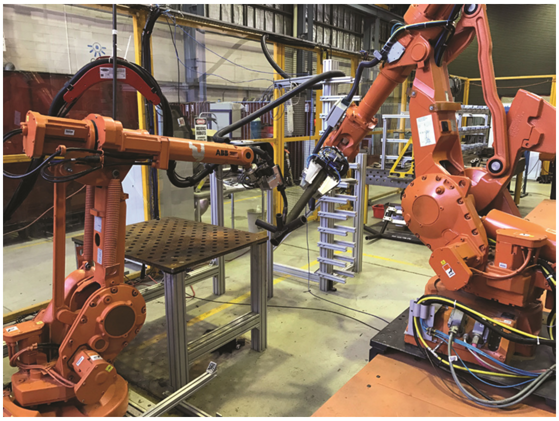


Fig. 1. The robotic welding cell with two 6-DOF manipulators

This paper presents the AOLP framework tailored specifically for arc welding. Previous papers have presented individual components in detail, but this paper aims to cover all the required stages for a full AOLP welding solution. Section 2 outlines the overall approach and Sects. 2.1 to 2.5 detail each stage. Results from a specific implementation are presented in Sect. 3 and conclusions are drawn in Sect. 4.

2 Automated Offline Programming

Automated Offline Programming (AOLP) uses planning algorithms to translate a CAD model directly into a robotic program with little human intervention. The process, introduced in [3, 5], is composed of several key stages as illustrated in Fig. 2. For the specific case of welding, they are:

(a) **3D CAD Model**
During product design, create a 3D CAD model of the parts to weld.
(b) **Process Definition**
Define the weld paths, orientations and key process parameters.
(c) **Process Planning**
Generate robot motions to perform the welding operation.
(d) **Trajectory Planning**
Plan intermediate motions to move between welds and home positions.
(e) **Calibration**
Use sensing to account for differences between the CAD model and the part.
(f) **Post Processing**
Optimize motions, and convert to robot code.

Steps (b)–(f) use algorithms to directly generate high quality welding programs with little or no human intervention. Sections 2.1–2.5 detail approaches to each of these stages.

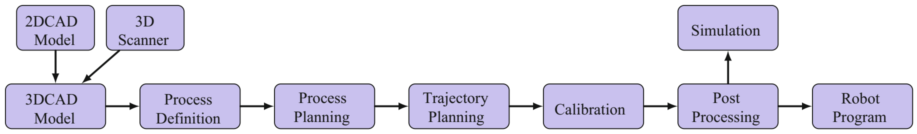


Fig. 2. The Automated Offline Programming (AOLP) process for welding

2.1 Process Definition

The first stage in the AOLP welding process is to define or extract the weld paths and necessary process parameters. This generates weld paths $p(t) \to SE(3)$, where $t \in [0,1]$ is the position along the weld. Several methods have been used to specify weld paths for CAD models [6]:

- Define weld paths within the CAD application.
- Pass data to an external program using an Application Programming Interface (API).
- Export CAD models in a universal format for processing in an external application.

Weld paths can be defined within the CAD application, such as the system demonstrated by Neto et al. [7] which uses line primitives in Autodesk Inventor to define welds. Reference [8] also used a similar approach to represent welds within the CAD model which were then extracted using the SolidWorks API. However, this approach relies on manual design work and is not fully automated. It is also difficult to deal with complex cases such as several curved work pieces joining together.

For direct programming, it is necessary to use semi-automated or fully automated methods to extract weld paths from CAD models. Semi-automated methods such as [9] prompt the user to select two plates to join and a straight weld is then generated. For a fully automated system, it is necessary to identify both straight and curved welds with no user interaction.

Automated Weld Path Generation. A fully automated system to identify welds between pairs of triangular meshes was presented in [10]. This is an ideal solution as it allows weld paths and normal to be automatically determined from complex and irregular geometry such as the example shown in Fig. 7. This method relies on repeatedly expanding and intersecting triangular meshes of each part, allowing both the weld paths and wire directions to be identified directly from a CAD model with no special metadata to define the weld.

2.2 Process Planning

The primary phase in the AOLP process is the generation of the robot motions to achieve each weld defined in Sect. 2.1. The weld path is only partially constrained, and there is freedom to tilt and rotate the torch to achieve a motion. In addition, the torch angles and workpiece position must be optimized to achieve a high-quality weld. This can be formulated as a task space (T-space) motion planning problem as shown in Table 1. T-space planning formulates the planning problem derived from the task being performed, as well as including task-specific constraints.

Table 1. T-space parameters describing the weld planning problem

Parameter	Description
t	Position along the weld path, $t \in [0,1]$
Rx,y,z	Rotation around the weld torch *x*, *y* and *z* axes. This corresponds to push, tilt and roll angles
ctwd	Contact Tip to Work Distance (CTWD)
configuration	Robot configuration representing which Inverse Kinematics (IK) solution to use
positioner	Configuration of any external robot or work piece positioners

There are several methods to solve T-space motion planning problems. A common method is to discretize the search space and use an algorithm such as A* [11] to generate a resolution-optimal solution. This approach does not work when the problem dimensionality is very high, as is common in robotic welding applications. A simple sampling-based planner implementation cannot be used as the probability of generating configuration space (*C*-space) samples which satisfy task constraints is very low.

Yao and Gupta [12] describe ATACE, which samples directly in *T*-space before converting samples to *C*-space. Shkolnik and Tedrake [13] sample *T*-space using a Rapidly exploring Random Tree, and then use a controller to grow the tree in C-space. Alternative, approaches such as [12] can repair samples to satisfy task constraints. Other algorithms include CBiRRT2 [14] and tangent-space sampling [15].

Positioner Configuration. One of the key *T*-space parameters to optimize weld quality is the configuration of the workpiece positioner. Positioners can take many forms, from rotary tables to full 6-DOF robots. Typically, each DOF in the positioner is discretized and candidate configurations are ranked using a cost function. This function is composed of several weighted weld metrics such as:

- Path incline or decline angle.
- Parent geometry profile angle. -Welding torch angle.

This cost function is typically adapted for the welding process being used. For example, with a typical Gas Metal Arc Welding (GMAW) process, the function would balance minimizing incline, maximizing parent geometry profile angle and minimizing welding torch angles (i.e. weld torch pointing down).

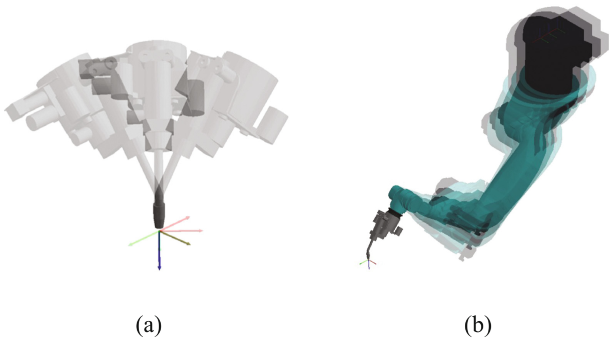


Fig. 3. Inherent redundancy in the (a) R_z and (b) *positioner* *T*-space parameters as shown in Table 1 [8]

Task Space Weld Planning. For welding, the T-space planning problem is formulated based on the weld path and weld gun angles, as shown in Table 1.

For a welding cell with a standard 6-DOF robotic manipulator and static work piece, the T-space planning problem also has six dimensions. The addition of any external robot axes or work piece positioners increases the problem dimensionality. As in [8] we use an A* [11] graph search to solve this problem while maintaining resolution optimal weld quality. However, the T-space problem is of sufficiently high dimensionality that a straightforward graph search is not possible.

In order to reduce the A* search to a tractable problem, the inherent redundancy of the T-space parameters is exploited as shown in Fig. 3. As can be seen, when changing the R_z torch roll angle, the position of the weld nozzle does not actually change. Another example is changing the *positioner* parameter: when this changes, the position of the entire weld torch does not change. This allows us to decouple the problem into smaller sub-problems which can be sequentially solved as in [8].

2.3 Trajectory Planning

Once the process motions have been generated, collision-free intermediate trajectories between home positions can be planned. In contrast to the T-space planning problem in Sect. 2.2, intermediate trajectories are planned in the robot's configuration space (*C*-space). A single robot configuration q is a vector of joint angles describing the robot's position, and the *C*-space C is the space encompassing all possible configurations. A portion $C_{obs} \subseteq C$ is occupied by obstacles, with the free space being defined as

$C_{free} = C - C_{obs}$. Given a start configuration q_{start} and goal q_{goal}, trajectory planning requires finding a path τ: $[0,1] \rightarrow C_{free}$ such $\tau(0) = q_{start}$, $\tau(1) = q_{goal}$ and any additional constraints are satisfied.

Sampling Based Planning (SBP) uses a graph structure of individual samples drawn from C to approximate the connectivity of the free space. This removes the need to explicitly represent C_{free} and C_{obs}, allowing for effective motion planning for high-DOF robots such as industrial robotic manipulators. The two most common SBPs are the Probabilistic Roadmap Method (PRM) [16] and Rapidly-exploring Random Trees (RRT) [17]. A PRM is a two stage multi-query planner: in the first stage a roadmap of random configurations connected by paths is generated and in the second stage this is queried to see if q_{start} and q_{goal} can be connected. In contrast, RRTs are single query planners which grow a tree from the start configuration q_{start} until q_{goal} is reached.

For an AOLP system, multi-query PRMs [16] are generally the most applicable approach as they benefit from repeated planning queries within the same work cell environment. An example roadmap generated within a manufacturing cell is shown in Fig. 4. There are many PRM variants such as Lazy-PRM [18] which can offer improved performance characteristics. The main challenge of using a PRM for automated welding is the need to plan into narrow passages, as the weld torch must be placed close to the work piece inside challenging areas. This can be addressed using sampling schemes suited to this scenario [19–21].

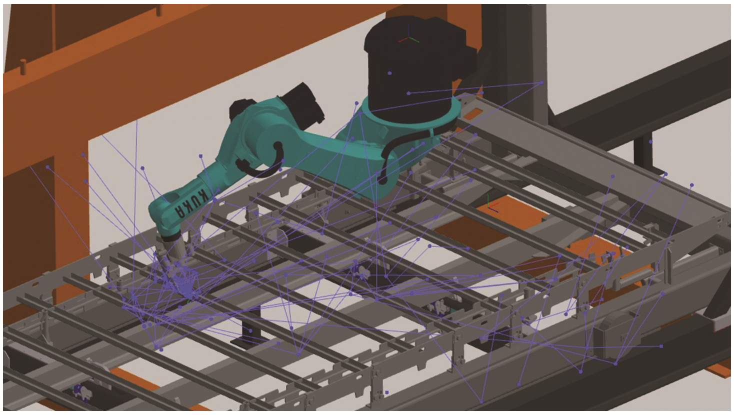


Fig. 4. A roadmap generated during trajectory planning [8]

2.4 Calibration

Calibration is necessary to adapt the weld path from the nominal CAD position to the "as-built" part position. The nominal weld path $p(t) \rightarrow SE(3)$ generated in Sect. 2.1 will differ from the "as-built" weld path $b(t) = p(t) + h(t)$ where $h(t)$ is the error function modelling the difference between the CAD model, the workpiece and robot pose error. Once the error is known the weld path can be adjusted during execution as shown in Fig. 5.

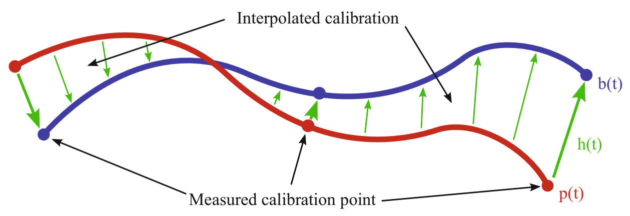


Fig. 5. A nominal CAD weld path $p(t)$, "as-built" path $b(t)$ and the measured error function $h(t)$ used for calibration

$h(t)$ is evaluated using sensors integrated in the robotic package. There are many commercially available solutions. Two common approaches are touch sensing, where the power source detects continuity between the welding system and the workpiece; and laser profilometry, where a 2D profile is evaluated by a dedicated processing system for a particular feature (such as a plane, corner, edge or bound). Other methods include structured light and computer vision [22]. Once the arc is established, seam tracking techniques [23, 24] can also be used.

Often it is not practical to continuously model $h(t)$, so discrete sensing actions are used to sample $h(t)$ and the intermediate values are linearly interpolated.

Also it is not always necessary to model all components of $SE(3)$ for the correction – this is often the case for corrections in the welding direction where an error will not lead to poor weld quality (with the exception of the start or end of a weld).

The online measurement of $h(t)$ involves the creation of sensing actions to be executed on the robotic system. These actions work by identifying feature positions and comparing them to known reference features from the CAD model. Creating these actions automatically involves careful consideration of the parent geometry, the effect of errors, and the sensing method used. We utilize a tiered search algorithm to determine the optimal robot actions to calibrate the weld position where sensing features are evaluated from most to least optimal:

1. Weld seam
2. Features on the parent parts of the weld
3. Nearby weld seams
4. Nearby part features

An important aspect of the calibration process is to evaluate the information gathered by sensing such that the weld can be calibrated in $SE(3)$. For example, additional information is required perpendicular to the welding direction to calibrate the weld start position.

Sensing motions and actions are planned as part of the AOLP process. These motions can be formulated as a T-space problem similar to Sect. 2.2.

Touch Sensing. Touch sensing is a common feature built into industrial welding power sources. It involves moving the robot along a linear path until an output from the power source indicates that continuity between the welding wire and the workpiece has been detected. Once continuity is detected, the robot motion is stopped and the position is recorded. Touch sensing requires several actions as each sensing operation provides a 1D calibration property.

Laser Profilometry. Laser profilometry typically employs a structured light mechanism and a camera to detect a 2D profile of the target. Common commercially available systems also provide a processing system to evaluate the profile and convert the raw scan data into simple metrics about the profile's type and position. Laser profilometry provides a 2D calibration property.

2.5 Post Processing

The final stage in the AOLP process involves sequencing, optimizing the generated motions and converting the process and trajectory plans to robot code. Sequencing can be based on minimizing cycle time, distributing heat input to minimize distortion, or other factors (Fig. 6).

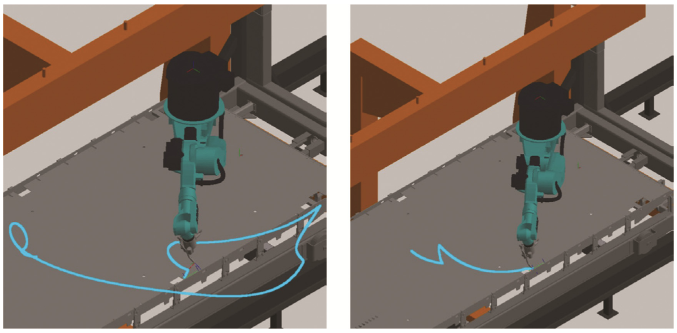


Fig. 6. An end-effector trajectory before (left) and after (right) optimisation [8]

Typical sampling-based planners, as outlined in Sect. 2.3, often produce sub-optimal paths and benefit from post-process optimization [25]. There are several techniques for post-process optimization which short-cut segments of the generated path. References [26, 27] attempt to skip points in the generated path to remove redundant motions. Sánchez and Latombe [28] attempt to skip entire randomly selected portions of the path at a time. Guernane et al. [29] presented a method which skips between the midpoints of path points and Hsu et al. [30] uses a similar technique which adds points to the path. Geraerts et al. [31] developed the partial shortcut method which applies these operations on one DOF at a time, and Polden et al. [25] extended this specifically for industrial robots to guide optimization towards DOFs which deviate the most from an ideal path.

Alternatively, planning variants can be used which embed optimization in the planning process. Algorithm variants such as PRM* and RRT* [32] converge to an optimal solution and can be used to minimize path length or other criteria during the trajectory planning phase. Therefore, post-process optimization may not be required.

As in [8] a path optimization procedure can also be used to generate motions between welds. A path between subsequent welds can be created by joining the path from the end of the first weld to the home position and the path from the home position to the start of the second weld. The optimization algorithm can then be used to shorten this non-optimal path. This allows acceptable quality intermediate motions to be generated without invoking a full planning algorithm as in Sect. 2.3.

The final step is to convert the generated process and trajectory motions into robot code and insert the relevant welding control commands. Common targets include ABB's RAPID language [33] or Kuka's KRL [34]. The conversion process from generated motions to a robot control program is generally straightforward.

3 Results

We present an AOLP implementation to weld pipe and plate geometry. This implementation utilizes two 6-DOF industrial robotic manipulators as shown in Fig. 1. An ABB IRB 4400 is used for workpiece positioning, and an ABB IRB 1400 is used for welding. Calibration sensing utilizes a ServoRobot DIGI-I/P laser profilometry scanner. For these results we present three scenarios as shown in Fig. 7:

1. a simple pipe-pipe structure,
2. a more complex pipe-plate structure, and
3. a complex multi-pipe node structure.

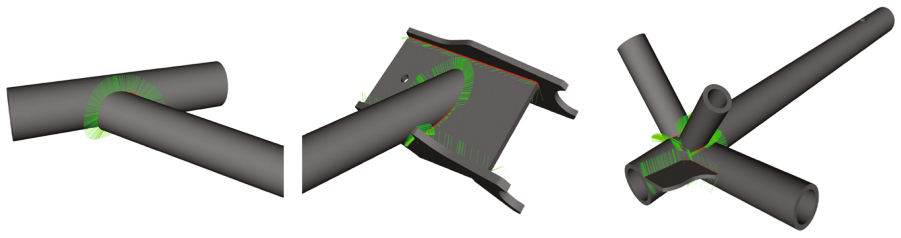

(a) Pipe-pipe (b) Pipe-plate (c) Pipe node

Fig. 7. Generated weld paths for each scenario

The first stage of the AOLP process is to generate the weld paths. The weld paths are then converted to robot motions using the following steps:

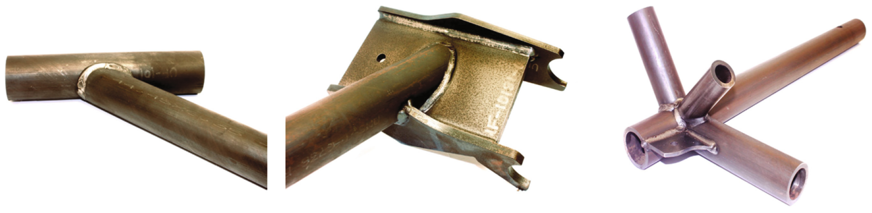
1. Workpiece positioning
2. Planning the weld path motion
3. Planning the sensing actions and motions

As the positioner is a 6-DOF robot we cannot naively sample all DOFs to evaluate workpiece positioner configurations. Instead we first rank combinations of (R_x, R_y) angles and then search within (R_z, x, y, z) for the best solution. Many of the welds in these scenarios are made up of complex curves where the weld seams cannot be measured directly for calibration purposes. Instead, the axes of the parent pipe objects are measured and used to create the error function $h(f)$. Welds are sequenced to ensure that weld starts are not placed on top of existing welds to improve the visual appearance of the fabrication.

The three scenarios shown in Fig. 7 were each planned on a MacBook Pro with a Core i7 CPU and 16 GB RAM. The overall planning time and a breakdown of the time taken for each stage is shown in Table 2. Each of the generated programs was then executed in the robotic welding work cell, resulting in high quality welds as shown in Fig. 8.

Table 2. Planning results for the presented AOLP implementation

	Pipe-pipe	Pipe-plate	Multi-pipe node
No. of bodies	2	4	5
No. of welds	4	11	17
Overall time (s)	153	711	793
Path generation (s)	0.539	1.22	2.22
Process planning (s)	104	506	595
Trajectory planning (s)	40.3	84.3	112
Post processing (s)	8.47	120	83.4



(a) Pipe-pipe (b) Pipe-plate (c) Pipe node

Fig. 8. Resulting welds for each scenario

The time taken to generate these programs is relatively short. The welding program for even the most scenario we tested was generated in under 15 min. This supports the applicability of this weld program generation for complex structures. The breakdown of planning time required for each stage in the AOLP process are as expected. Process planning, which includes generating the weld path motions and calibration positions, requires the majority of the processing time. We consider this to be an artifact of our optimization approach which uses a greedy search algorithm to maximize the final weld quality.

4 Conclusion

Automated Offline Programming enables truly flexible robotic welding capability. We listed the key steps of an AOLP process and explored the underlying technologies and algorithms. We then detailed our specific implementation that automatically generates weld programs for a wide range of simple and complex geometries, and tested the resulting programs on a robotic welding setup. Our results validate the approach and show the ability of this programming method to produce high quality welding results without human programming effort.

Future developments include applying automated programming techniques to more complex tasks such as integrated pick-and-place, assembly and welding. A limitation of the current approach is the requirement for a CAD model – another extension is to integrate part sensing so a CAD model is not required.

Acknowledgements. The authors would like to 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. The authors would also like to thank industry partners Thales Australia and Ekebol for their ongoing support and welding advice and expertise.

References

1. International Federation of Robotics (2011) World robotics 2011 - industrial robots. https://ifr.org/worldrobotics. Accessed 21 May 2014
2. Biggs G et al (2003) A survey of robot programming systems. In: Roberts J, Wyeth G (eds) Australia Australasian conference on robotics and automation, Brisbane
3. Pan Z, Polden J, Larkin N et al (2012) Recent progress on programming methods for industrial robots. Robot Comput Integr Manuf 28(2):87–94
4. Norberto PJ, Veiga G, Araújo R (2009) Programming-by-demonstration in the coworker scenario for SMEs. Indus Robot 36(1):73–83
5. Pan Z, Polden J, Larkin N et al (2011) Automated offline programming for robotic welding system with high degree of freedoms. Adv Comput Commun Control Autom 121:685–692
6. Foit K, Ćwikła G (2017) The CAD drawing as a source of data for robot programming purposes – a review. MATEC Web Conf 94:05002
7. Neto P, Mendes N, Araújo R et al (2012) High level robot programming based on CAD: dealing with unpredictable environments. Indus Robot 39(3):294–303
8. Larkin N et al (2016) Automatic program generation for welding robots from CAD. In: 2016 IEEE international conference on advanced intelligent mechatronics. IEEE, Banff, pp 560–565 (2016)
9. Polden J, Pan Z, Larkin N et al (2010) Offline programming for a complex welding system using DELMIA automation. Lect Notes Electr Eng 88:341–349
10. Hart P, Nilsson N, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107
11. Larkin N et al (2017) Automatic weld path generation for mesh objects. In: 2017 IEEE international conference on CYBER technology in automation, control, and intelligent systems. IEEE (in press)
12. Yao Z, Gupta K (2007) Path planning with general end-effector constraints. Robot Autonom Syst 55(4):316–327
13. Shkolnik A et al (2009) Path planning in 1000+dimensions using a task-space Voronoi bias. In: 2009 IEEE international conference on robotics and automation. IEEE, Kobe, pp 2061–2067
14. Berenson D, Srinivasa S, Kuffner J (2011) Task space regions: a framework for pose-constrained manipulation planning. Int J Robot Res 30(12):1435–1460

15. Stilman M (2007) Task constrained motion planning in robot joint space. In: 2007 IEEE/RSJ international conference on intelligent robots and systems. IEEE, Girod, pp 3074 – 3081
16. Kavraki LE, Svestka P, Latombe JC et al (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 12(4):566–580
17. Lavalle SM (1998) Rapidly-exploring random trees: a new tool for path planning. Algorithmic Comput Robot New Dir 1998:293–308
18. Bohlin R, Kavraki LE (2000) Path planning using lazy PRM. In: IEEE international conference on robotics and automation. IEEE, San Francisco, pp 521–528
19. Amato NM, Bayazit OB, Dale LK et al (1998) OBPRM: an obstacle-based PRM for 3D workspaces. Found Robotics 1998:1
20. Polden J, Pan Z, Larkin N et al (2013) Path planning with a lazy significant edge algorithm (LSEA). Int J Adv Rob Syst 10(4):198
21. Siméon T, Laumond J, Nissoux C (2000) Visibility-based probabilistic roadmaps for motion planning. Adv Robot 14(6):477–493
22. Agapakis J, Katz J, Friedman J et al (1990) Vision-aided robotic welding: an approach and a flexible implementation. Int J Robot Res 9(5):17–34
23. Kim JS, Son YT, Cho KS et al (1995) A robust method for vision-based seam tracking in robotic arc welding. In: 1995 IEEE international symposium on intelligent control. IEEE, Monterey, pp 363–368
24. Tao J et al (1999) Assessment of feedback variables for through the arc seam tracking in robotic gas metal arc welding. In: Proceedings 1999 IEEE International Conference on Robotics and Automation, vol4. IEEE, Detroit, pp 3050–3052
25. Polden J, Pan Z, Larkin N et al (2017) Adaptive partial shortcuts: path optimization for industrial robotics. J Intell Rob Syst 86(1):35–47
26. Isto P (2002) Constructing probabilistic roadmaps with powerful local planning and path optimization. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol 3. IEEE, Lausanne, pp 2323–2328
27. Kallmann M et al (2003) Planning collision-free reaching motions for interactive object manipulation and grasping. In: Computer graphics forum, vol 22. ACM, Los Angeles, California, pp 313–322
28. Sánchez G, Latombe JC (2003) A single-query bi-directional probabilistic roadmap planner with lazy collision checking. Robot Res 2003:403–417
29. Guernane R et al (2005) A smoothing strategy for PRM paths application to six-axes MOTOMAN SV3X manipulator. In: 2005 IEEE international conference on intelligent robots and systems. IEEE, Edmonton, pp 4155–4160
30. Hsu D et al (1999) Placing a robot manipulator amid obstacles for optimized execution. In: Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning. IEEE, Porto, Portugal, pp 280–285
31. Geraerts R et al (2004) Clearance based path optimization for motion planning. In: 2004 IEEE international conference on robotics and automation, proceedings, vol 3. IEEE, New Orleans, pp 2386–2392
32. Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. Int J Robot Res 30(7):846–894
33. ABB (2010) RAPID instructions, functions and data types. https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf. Accessed 21 May 2004
34. Kuka (2003) Software KR C2/KR C3 expert programming. http://forum.robotsinarchitecture.org. Accessed 21 May 2006