2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)
Banff, Alberta, Canada, July 12–15, 2016

Automatic Program Generation for Welding Robots from CAD

Nathan Larkin, Andrew Short, Zengxi Pan and Stephen van Duin

Abstract— **Industrial robotic automation is a key tool for manufacturing companies to achieve flexibility and low production costs. However, the cost associated with re-programming limits the economic viability for low production volumes. Automated Offline Programming is an approach that uses software algorithms to generate robot programs with little or no human effort. This contrasts with typical programming methods that require considerable human effort from highly skilled operators. This paper presents an Automated Offline Programming solution developed for a steel fabrication company and details the motion planning algorithms. It also describes a novel technique to decompose the welding path motion planning problem into sequential sub-problems such that greedy search techniques can be employed. The results show that this programming approach is effective for robot welding applications and reduces programming effort to effectively no cost.**

I. Introduction

In an era of globalisation it is critical for fabrication companies to maintain flexibility and low production costs in order to compete in the global marketplace. Industrial robot based automation is a proven solution to reduce labour overhead in the automotive industry, where the cycle time of each manufacturing station is short and batch size is high. However for low production volumes, the cost of retooling and reprogramming limits the economic viability.

Of all robotic applications, arc welding is a process of particular interest. In 2011 12.7% of the robots shipped were used in arc welding applications [1]. Retooling costs in robotic arc welding are inherently similar to those required in manual arc welding, particularly where product is manually tacked in position prior to full welding. This places an emphasis on the programming costs in this application.

A common method of programming for commercial industrial robots is the teach-repeat or lead-through method where the robot is manually moved to the desired positions and orientations for the task. Described in [2], these are recorded into the robot controller and repeated during program execution. [2] also details an Offline Programming (OLP) approach where programs are generated from Computer Aided Design (CAD) data and goes onto list the current gaps in available software to produce robot programs with low human effort and low cost.

In [3] a process called Automated Offline Programming (AOLP) is introduced where the complete OLP process is automatically completed. In [3] MATLAB based software processes CAD data to generate programs for a complex 13 Degrees Of Freedom (DOF) robot manipulator. While this work proves the feasibility of AOLP for arc welding applications, the programs generated still require a commissioning phase with considerable human effort.

Nathan Larkin, Andrew Short, Zengxi Pan and Stephen van Duin are with the School of Mechanical, Materials and Mechatronic Engineering at the University of Wollongong, Wollongong 2500, AU.

This paper presents an AOLP software system developed for a Small/Medium Enterprise (SME) involved in steel fabrication. It builds on the work presented in [3] detailing the approaches used to solve the motion-planning problem and introduces a novel method to plan the welding torch tool path which generates an optimal result. It also describes the underlying software library developed and critical functions such as calibration and code translation that improve the quality of programs generated such that they do not require any extra effort to implement and commission.

II. Related Work

The AOLP process for robotic arc welding is an integration of a number of underlying algorithms and systems. In this case we will specifically detail two problems:

1) Planning optimised tool paths within welding constraints, more commonly known as Task-space (T-space) motion planning, and
2) planning robot motions from home positions and between weld start and end locations, considered a Configuration-space (C-space) motion-planning problem.

A. Task-Space Motion Planning

T-Space represents a planning space with parameters derived from the robot task being performed, and incorporates task-specific constraints placed on the robot. An example is planning a pick-and-place operation using the position of the object being moved, rather than the C-Space joint angles of the robot. A common T-space formulation is the pose of the robot's end-effector.

There are many approaches that can be taken to solve T-space motion planning problems. Yao and Gupta [4] describe an approach that samples robot configurations directly in T-space before conversion to a C-space roadmap that is traversed using a constrained local planner. Shkolnik and Tedrake [5] take an approach that samples T-space into an RRT structure. They then apply a feedback controller to grow the tree in C-space.

Techniques can also be used to repair samples so they satisfy the constraints describing the T-space. Yao and Gupta [4] adapted Randomized Gradient Descent inside a Probabilistic Roadmap framework to satisfy task constraints. The CBiRRT2 algorithm, developed by Berenson [6], alternates between constrained C-space exploration and direct T-space

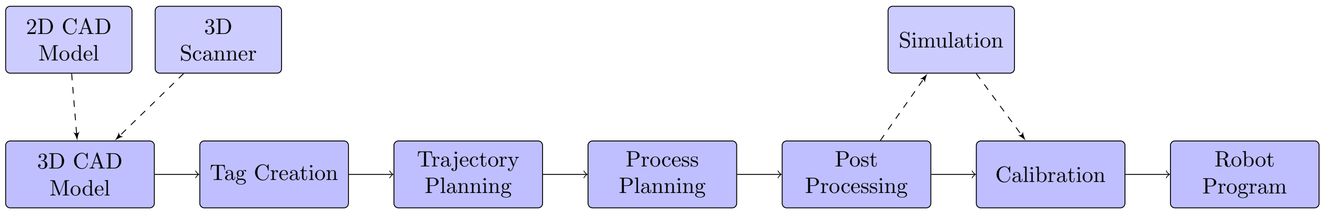


Fig. 1. Block diagram of the OLP process from [2].

sampling. Stilman [7] used tangent-space sampling and first-order retraction to constrain C-space sampling to T-space.

The application of these T-space algorithms for robotic arc welding is problematic. These approaches do find a solution that complies with the parameter constraints, however, the solution is unlikely to be the most optimal, resulting in poor weld quality. Conversely, if the parameter constraints are simply tightened around the most optimal values, the likelihood of finding a valid solution becomes low.

B. Configuration-Space Motion Planning

Robot motion planning for high DOF systems (such as robotic manipulators) is a well studied problem. Planning algorithms must generate a path from the robot's start configuration q_{start} to a goal q_{goal} which lies within the collision-free C-space region C_{free} and satisfies any additional constraints. The high dimensionality of such systems mean that explicit techniques are not feasible, so sampling-based planners are typically employed. The state of the art in sampling based planners was recently reviewed by Elbanhawi [8].

There are two major frameworks for sampling-based motion planning. The Probabilistic Roadmap (PRM) approach generates configuration samples and connects them using a local planner to form a roadmap. The start and goal configurations are then added to the roadmap and a graph search is used to see if a path can be generated [9]. In contrast Rapidly Exploring Random Trees (RRTs) grow a tree from the start configuration by generating random samples and extending the tree towards them [10]. As our use case required many planning operations in the same environment, we used the PRM framework as RRTs are generally only used for single-query planning.

The aim of this work is not to make significant contribution to motion planning, but to apply these algorithms in software to enable automation for arc welding with small batch volumes. We do however describe a new approach to solving the T-space motion-planning problem to generate the most optimal solution in a weld quality sense.

III. The Offline Programming Process

A detailed explanation of the OLP process can be found in [2]. Fig. 1 is a block diagram showing the typical process. The main steps are:

1) 3D computer model generation, typically generated during the design stages of a product.
2) Tag creation, where process start and end points are identified.
3) Trajectory planning, where robot motions are planned and the reachability and potential for collision of a robot is assessed.
4) Process planning, where each individual process is sequenced and optimised.
5) Post processing, where the required process I/O is added and the program is converted into the native programming language of the robot.
6) Calibration, run during program execution, where differences between the CAD data and the real world fabrication are measured and compensated for.

In an AOLP system, steps 2–6 are completed automatically by software algorithms such that programs are generated with minimal human effort and cost. This section will examine each key step in detail.

A. The Automated Offline Programming System

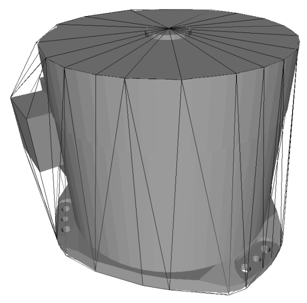
The underlying components of the AOLP system are:

1) The forward/inverse kinematic calculations,
2) the robot, workpiece and work cell geometry representation,
3) geometry position manipulation, and
4) the collision detection algorithm.

The robot kinematic calculations have been adapted from the previous work in [3] where closed form symbolic equations have been formulated for all possible robot configurations. The list of robots with inverse kinematic solutions has since been expanded to include the ABB IRB120, IRB1410, IRB2600ID, IRB6660, Epson C3 and Kuka KR60-3.

A software package built upon open source libraries [11], [12] was developed to handle rigid body and mechanism simulation and to detect collisions. Robot links, the work piece and robot cell are represented by a collection of triangular meshes and primitive shapes. The accuracy of the meshes is varied depending on the location of the component. For example, the robot tool models are more detailed than the base joints (1–3) as they are more likely to be in collision and an inaccurate model will more likely lead to real world collision or invalidate possible solutions. Convex hull shapes are used where possible to maximise the execution speed of collision checks. Fig. 2 shows the difference between collision model accuracy used for the robot base and tool model.

To account for the slight differences between the CAD model and the real world parts, the robot collision mesh


(a)

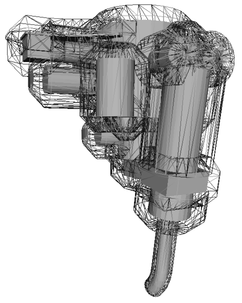
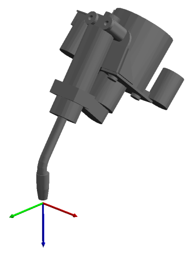
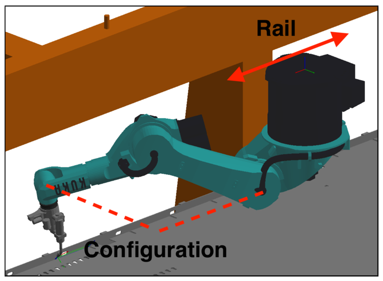

(b)

Fig. 2. Difference between collision model resolutions, shown as a wire mesh for the Kuka KR60-3 robot, where (a) is the base link and (b) is the welding torch.


(a)



(b)

Fig. 3. T-space parameters composed of (a) the co-ordinate frame attached to the end of the torch and (b) the robot configuration and rail position.

is expanded to provide additional clearance. The expansion distance is modified depending on the component and the planning problem. Planning a path from the home position to the start of a weld for example will use a large expansion value as there is little cost in terms of computational effort or path quality. When planning welds however, as the path will inherently be close to obstacles, a very low value is used, particularly for the welding torch collision model.

B. CAD Model and Tag Creation

There are many competing data formats for CAD data. In our case we chose the common STereoLithography (STL) mesh format due to ease of use and compatibility. The STL format represents object surfaces as a triangular mesh. This has some limitations as part information is lost (e.g. part thickness) and curved components are described with small errors. Much of this can be recovered using analysis algorithms if required.

Tags, or process points (in this case weld paths) are extracted from the CAD model. In this application, part intersections are not fully welded and control over the position, length and type of weld was controlled by the end user. Weld paths are represented as lines with a name representing the weld order, weld type and required procedure.

C. Path Planning

The process of path planning encompasses the generation of all robot paths. This involves two problems that require differing approaches:

1) Tool path planning, where the motion is constrained to the weld path with certain tool geometry, and
2) motion planning, where the motion is not constrained other than having a start and goal.

In both cases the motion planned must be feasible by the robot manipulator and cannot create a collision.

1) Tool Path Planning: Tool path planning for welding is formulated as a T-space motion planning problem as the path has workspace constraints such as the actual desired weld path and the weld gun angles specified in the weld procedure. We describe the welding task space with the following parameters, shown in Fig. 3a:

t	the position along the weld, $t \in [0, 1]$.
$R_{x,y,z}$	rotation around the X, Y, and Z axes of the tool tip
$ctwd$	the Contact Tip To Work Distance

In addition to these parameters on the torch, the path must have a continuous robot motion requiring all point solutions to have a common robot configuration. Additionally, the robot is mounted onto a linear rail that cannot be moved during welding. This adds two more parameters as shown in Figure 3b:

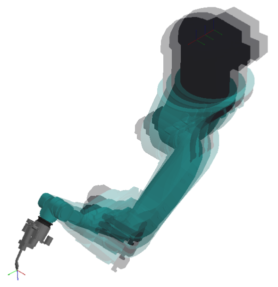
cfg	The robot configuration, $cfg \in \{1, 2, ..., 8\}$ representing the inverse kinematics solution.
$rail$	The position along the external linear axis connected to the base of the robot.

This results in the following 7-DOF T-space problem: *Given a weld path, determine a reachable, collision free and continuous path* $\forall t \in [0, 1]$ *that satisfies the T-space constraints.*

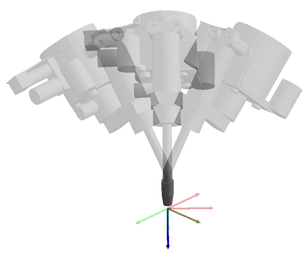
Where tool path planning for welding differs from a typical T-space motion-planning problem is the requirement for a high quality path. Although the tool angles allow a certain range to access difficult geometry (e.g. welding into a corner), there is often an optimal value that will result in the best weld quality and it is critical to determine the weld path with highest quality. This requirement has not been addressed by the approaches in literature.

To find the most optimal solution, an A* [13] search algorithm was adapted to search the T-space. As a result, this ensures that the highest quality path possible is found when coupled with appropriate cost functions for each parameter. While the A* algorithm is geared to finding the optimal solution through a graph, it is too computationally expensive to use for high DOF problems [8]. We subsequently decompose the main problem into several sub problems, with lower complexity. When these sub-problems are solved sequentially they form a solution to the overall problem.

We are able to decompose the main problem by splitting the kinematic chain into smaller components such that independent T-space parameters can be initially ignored. For example take the welding torch (Fig. 4a), the position of the welding torch does not change as the rail or the configuration of the robot changes and therefore, can be ignored. We go one step further and take into account object symmetry:



(a) Welding torch with varying rail



(b) Gas shroud with varying R_z

Fig. 4. Redundancy between T-space parameters and workspace objects.

TABLE I

DECOMPOSED T-SPACE SEARCH PROBLEM FOR TOOL PATH PLANNING IN ARC WELDING APPLICATION.

Parameter	Collision Object	Complexity
R_x, R_y, $ctwd$	Gas shroud	3 DOF
R_z	Welding torch	1 DOF
cfg, $rail$	Robot	2 DOF

TABLE II

WELDING SPECIFICATION IN THE FORM OF T-SPACE PARAMETERS, SHOWING MINIMUM, MAXIMUM AND OPTIMUM VALUES.

Parameter	Minimum	Maximum	Optimum	Discretisation Step
R_x	$-15°$	$15°$	$0°$	$5°$
R_y	$-30°$	$40°$	$20°$	$5°$
R_z	none	none	none	$10°$
$ctwd$	10mm	25mm	15mm	5mm
cfg	1	8	none	1
$rail$	0mm	12000mm	none	1000mm

the gas shroud is symmetric around the welding wire axis, and when rotated around this axis the effect on collision is unchanged (Fig. 4b). This means that for all rotations around the tool z-axis, the gas shroud collision will have the same result and do not require testing. The resulting decomposed problem is shown in Table I. As noted in the table, the overall motion planning problem has been decomposed into three sub-problems with a maximum dimensionality of 3. We will detail the formal T-space decomposition algorithm in a future paper.

As a consequence of the A* search algorithm, each parameter requires a discrete set of possible values. Typical discretisation values used in this application are set out in Table II. The parameter t is sampled in discrete increments along the weld path and a local planner checks connections between components such that change rate limits can be placed on the other T-space parameters.

The advantage of the decomposed problem is the ability to invalidate nodes at a high level with less collision components. All solutions that are a subset of the invalid node are no-longer possible valid solutions and are ignored. For example, using the T-space parameters and step sizes listed in Table II, each R_x, R_y and $ctwd$ combination that is invalidated allows 3744 possible fully specified candidate solutions to be also invalidated.

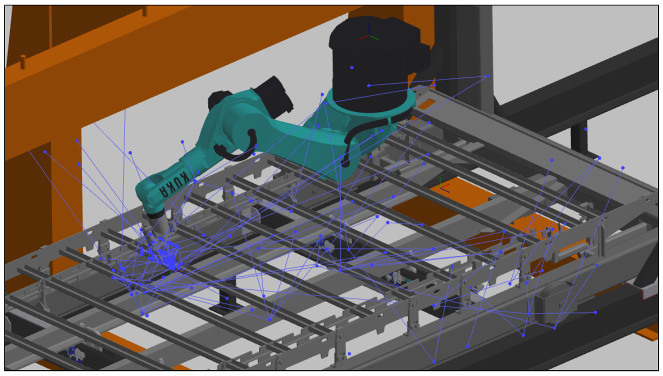


Fig. 5. A fully populated PRM roadmap showing sample density around the weld start.

2) Motion Planning: A Probabilistic RoadMap (PRM) motion planner was used to plan transition motions – motions to and from the home position and those between welds. As the robot had a discrete positions along a linear rail, separate roadmaps for each position were maintained to avoid duplicating roadmap information. The traditional PRM algorithm [9] has a separate roadmap construction and query phase. In contrast, we interleaved roadmap construction and querying to allow straightforward problems to be rapidly solved without fully populating the roadmap. Figure 5 shows a resulting roadmap generated by the system.

The choice of sampling strategy is key to the success of the PRM planner. In the case of robotic arc welding, it was observed that narrow passage type obstacle geometry exists around the start and goal position, but the intermediate space was free. Three strategies were used to efficiently plan in this environment:

1) A gaussian sampling strategy [14] was used, where pairs of close samples are generated. If one is in collision and the other free, the valid sample is added to the roadmap. This ensures that sampling is performed close to obstacles.
2) A number of "assistant points" were added close to the beginning and end of each weld. These were points that were generated in Cartesian coordinates and mapped to C-space if collision free.
3) Sampling is initially biased to be close to the start and goal positions until they are well connected to the roadmap (i.e. are directly or indirectly connected to n vertices).

The effect of this sampling strategy can be seen in Figure 5. Most samples are clustered around the robot's home position and the goal configuration, with fewer samples in the free space as desired.

To demonstrate the performance of the PRM planner, we ran it on 250 representative planning problems from the manufacturing setup. The PRM successfully found valid paths for all problems, which was a key requirement. Most

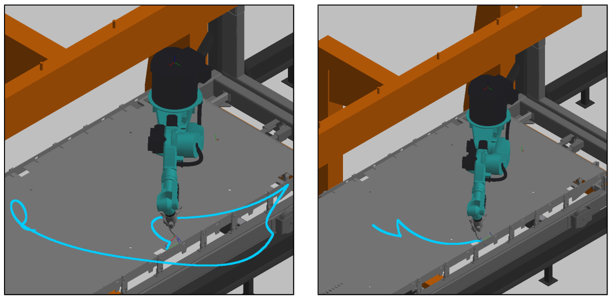
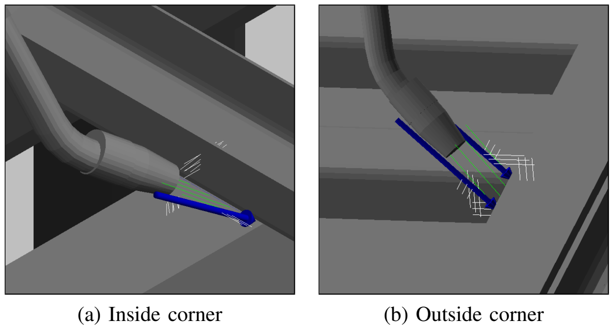


Fig. 6. A generated path before and after smoothing.


(a) Inside corner (b) Outside corner

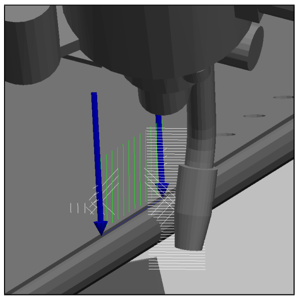

(c) Side touch

Fig. 7. Available touch sense operations.

problems were solved rapidly, with 83% of paths planned in under a second. However there were several paths which took significantly longer due to challenging geometry, with the longest taking 56s. The median time was 0.221s and the mean 1.64s, with a standard deviation of 5.92s.

Further optimisations could be done on the PRM planner to improve performance such as using a Lazy PRM variant [15] or using a PRM which can adapt high quality motion primitives [16], since most generated transition paths are similar. However, for this case the performance was acceptable.

3) Path Smoothing: The quality of the paths generated by a PRM planner is generally quite low, with jerky and unnecessary motions as shown in Figure 6. A shortcutting algorithm was applied using the techniques from [17], yielding shorter and higher quality paths. This was iteratively run until the path improvement was negligible.

The shortcutting procedure was also used to generate more efficient paths in between welds. If possible, a path between subsequent welds was created by concatenating the generated path from the weld end to robot home to the path to the start of the next weld. The shortcutting algorithm was then applied to optimise this path. This yielded efficient intermediate paths, reducing cycle time while not requiring the significant additional time required to invoke the PRM planner again.

D. Calibration

Touch sensing is used to locally calibrate each weld. Touch sensing places the welding system in a state that detects when an electrical connection is made between the welding wire or gas shroud to the workpiece. When connected to the robot, this can be used to determine the actual plate position and compensate accordingly. To minimise robot motions to cut and prepare the welding wire, the gas shroud is used as the sensing position. During touch sensing the wire is simply retracted into to torch to prevent erroneous readings.

There are a number of touch sense types supported in this system, as shown in Fig. 7:

1) Inside corner, the plates associated with the weld have an opening less than 180°.
2) Outside corner, the plates associated with the weld have an opening greater than 180°.
3) Side touch, the touch position is not on flat material and the side of the gas shroud should be presented to the work piece.

Touch sense motions are generated with a number of ray casts, torch collision checks and geometry calculations to ensure that the object being sensed can be used as a calibration location. The nominal calibration values are determined from the CAD data and used as a reference in the robot program when executing.

E. Weld Ordering

The weld order was mainly left to the end user as requested. Some optimisations were made to group welds with common linear rail positions to improve cycle time.

F. Robot Code Conversion

In this application we utilised a common program for all weld paths. The raw output of the path planner is simply a list of robot motions. There is a list of motions to move from the home position to a weld, a list of motions to perform a weld and a list of motions to return home or move to the next weld. The common program simply accesses the data list generated for the current weld and executes. This philosophy allows for global changes required for each weld to be made into one program only and can be applied to previously generated motion lists without re-generation.

The code translator creates the motion list by converting the positions into the joint or Cartesian poses as specified by the robot manufacturer. For Cartesian poses the robot configuration data is also added. Currently code translators exist for ABB, Epson, Kawasaki and Kuka robots.

IV. Results

To characterise the performance of the system, the planner was run on five representative weld groups, each of which contained 20 welds. The robot manipulator was composed of

TABLE III

WELD PLANNING SUCCESS RATE AND TIME TAKEN.

Weld Set	Planned	Total Time	T-Space Time	C-space Time
1	100%	25.9s	0.89s	24.8s
2	100%	3.50m	2.65m	1.31m
3	80%	14.56m	10.6m	2.78m
4	100%	56.6s	16.6s	39.1s
5	100%	27.0s	1.10s	22.5s

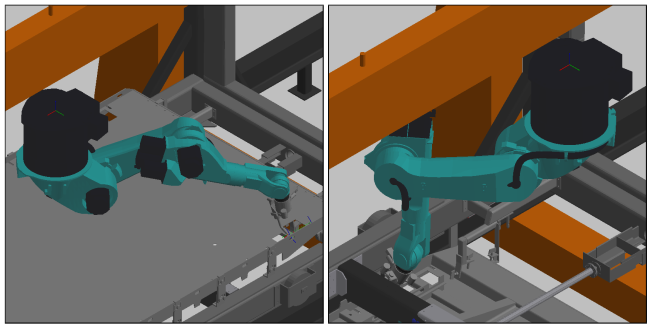


Fig. 8. Two representative weld problems showing varying levels of difficulty where (a) is unobstructed and (b) is in a narrow passage.

7500 triangles, the work pieces of between 6300 and 22000 triangles and there were other collision objects present in the scene. Each weld group was of different relative difficulty, as shown in Fig. 8. Tests were run in a virtualised Windows 8.1 instance on a Macbook Pro with a 4-core 3.1 GHz i7 CPU and 16GB RAM. The application was not optimised to take advantage of multiple cores for planning.

As can be seen from the results in Table III, the system is capable of planning welds rapidly enough to be directly integrated into the manufacturing process. The average time taken per weld for the planning process to complete was 24.9s. There was high variation in the planning times due to different relative difficulties and the greedy tool path planning algorithm used. Additionally there was a small number of planning failures, that when inspected, were infeasible to weld. These results show that generating a robot program for a given work piece is effectively a zero-cost operation and demonstrates the effectiveness of using AOLP in welding applications for low volume manufacturing.

V. CONCLUSION

This paper describes a method to automatically generate robot programs from welding directly from CAD data. The programs generated are ready to run and do not require prior commissioning. Programs are generated by the end user and are completed with less than one minute of human interaction. Robot programs generated have proven to be high quality, require no commissioning, and result in robot motions and welds which satisfy customer quality requirements.

AOLP technology is critical into reducing the cost of joining materials with arc welding processes where batch volumes are low and allow SMEs with high labour cost economies compete in the global marketplace.

VI. ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of the Defence Material Technology Centre (DMTC), which was established and is supported by the Australian Governments Defence Future Capability Technology Centre (DFCTC) initiative. The support from industry partner, Ekebol, is also appreciated. Andrew Short is supported by the Australian Postgraduate Award (APA) and the University of Wollongong Global Challenges program.

REFERENCES

[1] I. F. of Robotics, "World robotics 2011 - industrial robots," Tech. Rep., 2011.

[2] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[3] Z. Pan, J. Polden, N. Larkin, S. van Duin, and J. Norrish, "Automated offline programming for robotic welding system with high degree of freedoms," in *Advances in Computer, Communication, Control and Automation*. Springer Berlin Heidelberg, 2011, pp. 685–692.

[4] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 316–327, 2007.

[5] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2061–2067.

[6] D. Berenson, S. Srinivasa, and J. Kuffner, "Task Space Regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[7] M. Stilman, "Task constrained motion planning in robot joint space," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 3074–3081.

[8] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Access, IEEE*, vol. 2, pp. 56–77, 2014.

[9] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[10] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep., 1998.

[11] E. Coumans *et al.*, "Bullet real-time physics simulation," http://bulletphysics.org, 2013.

[12] "OGRE 3D Graphics Engine," http://www.ogre3d.org/, 2013.

[13] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[14] V. Boor, M. H. Overmars, V. der Stappen, and A. Frank, "The gaussian sampling strategy for probabilistic roadmap planners," in *Robotics and Automation, 1999. Processdings. 1999 IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1018–1023.

[15] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 521–528.

[16] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," in *Algorithmic foundation of robotics VII*. Springer, 2008, pp. 507–522.

[17] J. Polden, A. Short, A. Visser, Z. Pan, N. Larkin, and S. van Duin, "Adaptive partial shortcuts: Path optimization for industrial robotics," Submitted, 2015.