Optimal trajectory planning of complicated robotic timber joints based on particle swarm optimization and an adaptive genetic algorithm

Yiping Meng¹ · Yiming Sun² · Wen-shao Chang¹

Received: 5 August 2020 / Accepted: 4 February 2021 / Published online: 11 April 2021
© The Author(s) 2021

Abstract

In this paper, a methodology for path distance and time synthetic optimal trajectory planning is described in order to improve the work efficiency of a robotic chainsaw when dealing with cutting complex timber joints. To demonstrate this approach one specific complicated timber joint is used as an example. The trajectory is interpolated in the joint space by using a quantic polynomial function which enables the trajectory to be constrained in the kinematic limits of velocity, acceleration, and jerk. The particle swarm optimization (PSO) is applied to optimize the path of all cutting surfaces of the timber joint in operating space to achieve the shortest path. Based on the optimal path, an adaptive genetic algorithm (AGA) is used to optimize the time interval of interpolation points of every joint to realize the time-optimal trajectory. The results of the simulation show that the PSO method shortens the distance of the trajectory and that the AGA algorithm reduces time intervals and helps to obtain smooth trajectories, validating the effectiveness and practicability of the two proposed methodology on path and time optimization for 6-DOF robots when used in cutting tasks.

Keywords Optimal trajectory · Time optimization · Robotic fabrication · Particle swarm optimization · Adaptive genetic algorithm · Polynomial interpolation · Chainsaw cutting

1 Introduction

During the past decades, the use of robotic fabrication practices has increased in many fields, especially in architecture. Robotic fabrication shows potential for improving processes across a variety of application, from improving the sustainability of additive manufacturing (Ford and Despeisse 2016), such as reduction in waste and energy saving (Faludi et al. 2015), to its use for the execution of both standardized and customized construction assembly tasks (Keating and

Oxman 2013). Robotic fabrication shows its potential benefits for complex structures in concrete (Agustí-Juan et al. 2017), lightweight timber plates through robotic milling combined with manual assembly work (Krieg et al. 2015), and brick laying using mobile robots (Helm et al. 2012). Compared to CNC technologies, which are used in timber prefabrication predominantly to manufacture standard components like beams and plates (Popovic et al. 2016; Eversmann et al. 2017), robotic fabrication shows its strength in mass customization of movements to make more complicated timber joints due to the potential for complex manipulator trajectories (Jeffers 2016). However, the complexity of the design makes the trajectory more intricate which often increases process time. Robotic chainsaw cutting has a range of applications in timber fabrication industries, however, to achieve an efficient process there are specific aspects of the task which must be taken into consideration. The surfaces in the cutting task are in different surfaces and the cutting order passing through these surfaces has significant impact on working efficiency. When dealing with complex joints, path planning for robotic chainsaw cutting can be

✉ Yiping Meng
ymeng16@sheffield.ac.uk

Yiming Sun
ysun67@sheffield.ac.uk

Wen-shao Chang
w.chang@sheffield.ac.uk

¹ People, Environments and Performance Group, School of Architecture, The University of Sheffield, Sheffield, UK

² Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK

time-consuming and manual creation of paths can not guarantee the most efficient solution.

To generate an efficient trajectory requires the satisfaction of specific objectives like time reduction and minimizing of the quick short movements of the robot manipulator. These quick short movements create sudden jerks to the end effector and can reduce the quality of the cut of damage the chainsaw. There are diverse researches on robotic automation construction (Labonnote et al. 2016) and various trajectory optimization techniques (Ata 2007; Jeevamalar and Ramabalan 2012; Ratiu and Prichici 2017), however research on specific trajectory planning optimization methodology applied on a full-scale architecture joint fabrication is rare. Therefore, this study takes the following case as an example to test the feasibility of the trajectory optimization techniques in real practice.

1.1 Case study

In this case, tree trunks are chosen to be the material which would be firstly scanned and then cut by a robot arm equipped with a chainsaw. This case study illustrates the problems within trajectory planning of complicated timber joints in order to test optimization methods for increasing process efficiency. The complicated timber joint was designed and fabricated in a workshop at the international conference: ROBIARCH 2018. The timber joints are designed in Rhino, and the simulation of cutting process using a KUKA robot equipped with a planar cutting tool (chainsaw) uses KUKA|prc. The details of the robotic chainsaw cutting task including the design of the timber joints and the setting work space are shown in Fig. 1.

The first parameter of the process is the shape of the cutting faces as the input surfaces. The cutting tool is always at

Fig. 1 Diagram of chainsaw cutting

the backside of the input surface, which could be achieved by reversing the normal direction of the input surfaces. This is to align the normal direction of the tool with the normal direction of input surfaces to be in the right offset side. The cutting direction and the cutting order of the four points within one cutting surface is consistent with the drawing order 1, 2, 3, 4, shown in Fig. 2. In this case, the surfaces that have similar orientation are grouped to reduce the amount of re-orientations to save time and avoid collisions.

When planning multiple cuts, the main difficulty is to designate the cutting order of different cutting surfaces to avoid kinematic singularities and at the same time to avoid collisions. The timber joint in this case has been divided into four cutting parts, and every part has more than 10 cutting surfaces; each surface has four cutting points. The cutting order combinations could be $n! \times C_4^3$, where n is the number of cutting surfaces, which means it is impractical to figure out every rational combination. Two parts of the timber joint have been fabricated in the work space (Fig. 3) and the results are shown in Fig. 4a, the same as the initial design in Rhino. During the workshop, the cutting order was set manually. The whole process took considerable time and decreased the degree of automation of robotic fabrication. This also added uncertainty as to whether the manually designed path was the shortest travel distance or not. To tackle the difficulty found during the workshop, one part of the joint (Fig. 4b) is selected to as an example to be optimized.

This study took 13 cutting surfaces of the first part as the example for addressing complicated robotic timber joints within trajectory planning and utilizes a multi-objective technique to obtain trajectory planning optimization

Fig. 2 Movement of chainsaw within one cutting surface

Fig. 3 On-site fabrication

(a) Results of workshop

(b) Example of the Case Study

Fig. 4 Timber joint fabrication

of robot manipulators corresponding to the specific example in this study, to put forward a feasible solution to find the best cutting order automatically to achieve the shortest distance and time.

1.2 Trajectory planning in robotics

The trajectory planning problem in robotics is to find a motion along the given geometric path, taking into account process constraints, to generate inputs for the control system of the manipulator, which ensures the implementation of the planned movement (Gasparetto and Zanutto 2010; Gasparetto et al. 2012). Trajectory planning in robotics is generally carried out in the operating space (Luh and Lin 1981) or joint space (Craig 2009). The trajectory described in the joint space returns the position for each joint of the manipulator, corresponding to the points set by the user.

Though in the operating space, the trajectory of the end-effector can be described directly, it is very difficult to solve the problem analytically due to the highly coupled and highly sub-linear characteristics of robot dynamics equations. In most cases, the trajectory planning is solved in joint space to avoid kinematic singularities or manipulator redundancy (Gasparetto et al. 2015). The most common method is to transform the trajectory function from operating space into joint space, and then to control the manipulator in the joint space.

The trajectories defined in the joint space are constructed by means of interpolation functions such as polynomial, spline, Bezier, and NURBS, etc., which are restricted to kinematic constraints. Conventional polynomial interpolation has developed 343,445 and novel 5455 methods based on different orders polynomial functions (Cook and Ho 1984; Petrincic and Kovacic 2007; Boscaroli et al. 2012). Trajectories interpolated by spline functions shows good performance ensuring the continuity of the acceleration or jerk (derivative of the acceleration), cubic B-spline, quantic B-spline as 7th-order B-spline have been tested to obtain trajectory smoothness (Ramabalan et al. 2009; Gasparetto and Zanutto 2008; Zhu et al. 2010).

Numerous techniques for optimal offline trajectory planning have been developed in the last decades by different researchers in the field. The trajectory planning algorithm takes the geometric path, kinematic and dynamic constraints as inputs and generates the trajectory of the joints, or of the end-effector by means of position, velocity and acceleration expressed in time sequence (Siciliano et al. 2010). The aim of the trajectory planning optimization is to improve the manufacturing productivity and the movement stability, which means the short execution time or smooth path (Huang et al. 2018). Most of the trajectory planning algorithm optimizations are based on an objective function of some parameters to improve: (1) minimum execution time, (2) minimum energy, and (3) minimum jerk, or a combination of these (Gasparetto et al. 2012).

1.3 Optimal techniques

The most common used trajectory planning optimization techniques are minimum-time algorithms, due to the need for increasing productivity in the industrial sector (Gasparetto and Zanotto 2010; Piazzoli and Visoli 1998). A convex optimization for time trajectory planning was put forward by converting time-optimal problem into convex optimal control problem, neglecting the viscous friction (Verschuere et al. 2009). When robot moved at high speed, time-optimal optimization could be non-convex taking the viscous friction of robots into consideration (Shen et al. 2019). To solve this inaccuracy, a new genetic algorithm for time-optimal and collision-free trajectory in complex environments was developed (Abu-Dakka et al. 2015). Also, these algorithms meet torque and acceleration discontinuity required by assuming that robot was a rigid body. To overcome these drawbacks, the actuator jerks were introduced as an additional metric for optimization (Constantinescu and Croft 2000).

Besides time optimization, jerk minimization results in better performance in tracking capacity (Gasparetto et al. 2015; Kyriakopoulos and Saridis 1988), being able to achieve smaller excitation of resonance frequencies (Gasparetto et al. 2015). Squared jerk has also been set as the optimization object using spline interpolation (Gasparetto and Zanotto 2007; Simon 1993; Zanotto et al. 2011).

Minimizing the energy consumption is another alternative optimization objectives, which leads to the smoothness of the trajectory with smaller stresses on the robot manipulator (Field and Stepanenko 1996). Cubic B-spline functions, fifth order B-spline and Lagrange interpolation have been applied to construction trajectories within energy minimization objective (Gasparetto and Zanotto 2007; Sato et al. 2007; Luo et al. 2015).

In addition, hybrid optimizations have also been proposed instead of a single optimization objective function. Time-energy optimal trajectories planning is composed by two terms, the first related to the travel time and the second to energy. The optimal objective is to reduce the stresses of the actuators and to promote the most efficient trajectory tracking (Balkan 1998; Shiller 1996). Similar to time-energy, time-jerk optimal trajectory planning also consists of two parts, time and jerk (third derivative of position) (Gasparetto and Zanotto 2007, 2008; Zanotto et al. 2011). The simulation and experiment results validated the effectiveness of the methods. However, the trade-off between the two criteria is controlled by two weights of the objective function which still requires artificial adjustment. To solve the multi-objective problems, there is an emerging alternative which is to employ evolutionary algorithms such as particle swarm optimization (PSO) and adaptive genetic algorithm (AGA) to solve the above-mentioned drawbacks (Ata and Myo 2005; Saravanan et al. 2008).

PSO was devised in 1995 inspired by the flock of birds' behaviour and has been applied to many fields of optimization problems (Eberhart and Kennedy 1995). As a random-search algorithm, studies have verified the convergence speed of particle swarms by offering the initial behaviour of a population (Reyes-Sierra and Coello 2006; Helwig and Wanka 2008; Trelea 2003). In addition to the performance of PSO on continuous optimization problems, many studies have tested the application of PSO on discrete problems (Kennedy and Eberhart 1997; Wang et al. 2003; Shi et al. 2007), including the optimization of path planning (Hoffmann et al. 2011). PSO shows its potential in finding the shortest collision-free path for mobile robots effectively (Zhao and Yan 2005). More simulation results shows the competence of PSO in path planning in static or dynamic environments (Xu et al. 2008; Hao et al. 2007). PSO could also tackle multi-objective robot path planning with uncertainty in planar surface (Zhang et al. 2013). Apart from travel distance, a multi-objective particle swarm optimization algorithm is applied to minimize the travel time, energy and the distance of the end-effector (Xu et al. 2015).

Genetic algorithm (GA), inspired by natural genetic systems and natural selection (Davis 1991), is viewed as another advantageous evolutionary process to optimize non-linear and complex problems which traditional method could not solve (Davis 1987). The AGA varied the probability of crossover and mutation probability depending on the fitness value of the objective function (Srinivas and Patnaik 1994). The AGA method significantly optimizes the time-intervals between each section of the trajectory so as to raise the efficiency of the robot (Liao et al. 2010). Combining the quintic polynomial function to interpolate the trajectory, time-optimal trajectory planning could be realized by AGA to reduce running time (Zhang et al. 2018). When tackling the same trajectory time-optimal problem, the simulation results show that AGA converge better than GA and reduce the probability of local best solutions to achieve better stability of robot (Fu and Ju 2011).

This study reports two optimal objectives (travel distance and execution time) to obtain smooth and short trajectories of robot manipulators. With respect to most path and time optimal algorithm techniques in scientific literature so far, the PSO along with AGA are selected to deal with the two optimization objectives, respectively.

This paper is organized as follows. In Sect. 2, the trajectory optimization problem is formulated and kinematic singularities are defined; fifth-order polynomial function is chosen to construct the joint trajectory, and Modified Denavit–Hartenberg is chosen to build the robot model. In Sect. 3, the robot arm model is built in MATLAB and the data of the model in Rhino is converted to interpolate the position, velocity and acceleration of the six joints according to the setting points. Section 4, describes the

whole algorithms in detail and describes the execution. Section 5 shows the results of the simulation.

2 Methodology

2.1 Formulation of the optimization objectives

The KUKA robot with six degrees of freedom is considered in this study. The trajectory planning methodology described in this paper assumes that the geometric path is converted from the specific geometry vertices. The path is given as a sequence of points in the operating space, which represent the positions and orientations of the end-effector. The points would be transformed to consecutive points in the joint space through inverse kinematics. Based on the points in the joint space, a trajectory would be constructed and then optimized for the minimization objectives. Obstacle avoidance and collision problems are assumed solved and will not be in the scope of this study.

Considering the uniqueness of this case study, the proposed technique does not set the geometric path as a variable, which is different from the current trajectory planning. The minimization of the travel distance of the end-effector has direct effect on the time minimization. In this study, two objective functions are formulated: the travel distance and the execution time.

The two optimization problems are mathematically defined as follows and the meaning of the symbols are listed in Table 1 and Table 2: Minimize:

Table 1 Meanings of symbols appearing in Eqs. (1) and (2)

Symbol	Meaning
D	Traveling distance objective function
L_i	Distance between two surfaces
$L_{i,j}$	Distance between the two vertices belonging to one surface
p_k	Position of the k th joint
x_e	x coordinate of the end-effector
y_e	y coordinate of the end-effector
z_e	z coordinate of the end-effector
U_j	Position range of the j th joint
U_x	Working space range for x coordinate
U_y	Working space range for y coordinate
U_z	Working space range for z coordinate
$ J[q] $	Determinant of Jacobian matrix
N	Number of surfaces
M	Number of vertices
n	Number of robotic manipulator joints

Table 2 Meanings of symbols appearing in Eqs. (3) and (4)

Symbol	Meaning
T	Traveling time objective function
h_i	Time interval between the i th via-point and $(i + 1)$ th via-point
t_i	i th time node
p_i	Position of the i th joint
$\dot{p}_i$	Velocity of the i th joint
$\ddot{p}_i$	Acceleration of the i th joint
$\dddot{p}_i$	Jerk of the i th joint
VC_i	Velocity constraint for the i th joint
WC_i	Acceleration constraint for the i th joint
JC_i	Jerk constraint for the i th joint
$n + 1$	Number of the via-points in the joint space
N	Number of robotic manipulator joints

$$D = \sum_{i=1}^N L_i + \sum_{i=1}^{N-3} \sum_{j=1}^M L_{i,j} \quad (1)$$

Subject to:

$$\begin{cases} p_k(t) \in U_j, k = 1, 2, \dots, n \\ x_e(t) \in U_x \\ y_e(t) \in U_y \\ z_e(t) \in U_z \\ |J[q]| \neq 0. \end{cases} \quad (2)$$

Minimize:

$$T = \sum_{i=1}^{n-1} h_i = \sum_{i=1}^{n-1} (t_{i+1} - t_i) \quad (3)$$

Subject to:

$$\begin{cases} |\dot{p}_i(t)| \leq VC_i, i = 1, 2, \dots, N \\ |\ddot{p}_i(t)| \leq WC_i, i = 1, 2, \dots, N \\ |\dddot{p}_i(t)| \leq JC_i, i = 1, 2, \dots, N. \end{cases} \quad (4)$$

The two objective functions are related. Objectives one would be optimized first to work out the fittest travel path and based on the optimal path travel time would be optimized to get the smooth trajectory. Also, the kinematic constraints for velocity, acceleration and jerk as well as the interpolation conditions for all via-points (vertices of the cutting surfaces) must be met.

2.2 Definition of the trajectory by means of fifth-order polynomial function

To solve the optimization problems above, fifth-order polynomial function is selected to construct the trajectory in joint space. $\theta_i(t)$ denotes the position of six joints in angle. According to the definition of the quantic polynomial function (Thomas et al. 2010), the trajectories of six joints are expressed mathematically as follows, i means the number of joints:

$$\theta_i(t) = a_{i0} + a_{i1}t + a_{i2}t^2 + a_{i3}t^3 + a_{i4}t^4 + a_{i5}t^5 \quad (5)$$

$$i = 1, 2, \dots, 6.$$

For one of the trajectories, the start time is set to be t_0 and end time to be t_f ; the start point and end point of position, velocity and acceleration constraints are expressed as follows:

$$\begin{cases} \theta_i(0) = \theta_{i0} \\ \theta_i(t_f) = \theta_{if} \end{cases} \quad (6)$$

$$\begin{cases} \dot{\theta}_i(0) = \dot{\theta}_{i0} \\ \dot{\theta}_i(t_f) = \dot{\theta}_{if} \end{cases} \quad (7)$$

$$\begin{cases} \ddot{\theta}_i(0) = \ddot{\theta}_{i0} \\ \ddot{\theta}_i(t_f) = \ddot{\theta}_{if} \end{cases} \quad (8)$$

$$\begin{cases} \dddot{\theta}_i(0) = \dddot{\theta}_{i0} \\ \dddot{\theta}_i(t_f) = \dddot{\theta}_{if} \end{cases} \quad (9)$$

Supposing the initial and ending positions of the 6-DOF robot are known, position constraints, velocity constraints, acceleration constraints of the initial and end points are also included. The 6 parameters of the fifth-degree polynomial uniquely determine a trajectory to meet the constraints. Deriving Eq. (5), the velocity expression $\dot{\theta}_i(t)$ of the trajectory is obtained as

$$\dot{\theta}_i(t) = a_{i1} + 2a_{i2}t + 3a_{i3}t^2 + 4a_{i4}t^3 + 5a_{i5}t^4 \quad (10)$$

$$i = 1, 2, \dots, 6.$$

Similarly, the acceleration function $\ddot{\theta}_i(t)$ and jerk function $\dddot{\theta}_i(t)$ are expressed as below as the second and third derivative of Eq. (5):

$$\ddot{\theta}_i(t) = 2a_{i2} + 6a_{i3}t + 12a_{i4}t^2 + 20a_{i5}t^3 \quad (11)$$

$$i = 1, 2, \dots, 6.$$

According to the coefficients, the equation of jerk (third derivative of position) could be formed to get the constraints of the JC_i .

$$\dddot{\theta}_i(t) = 6a_{i3} + 24a_{i4}t + 60a_{i5}t^2 \quad (12)$$

$$i = 1, 2, \dots, 6.$$

According to Eqs. (6)–(9), the constants of a_{ij} could be conducted, means the j th constant of the i th joint, $i = 1, 2, \dots, 6$, means the number of joint and $j = 0, 1, \dots, 5$, means the index of the constants a :

$$\begin{cases} a_{i0} = \theta_{i0} \\ a_{i1} = \dot{\theta}_{i0} \\ a_{i2} = \frac{\ddot{\theta}_{i0}}{2} \\ a_{i3} = \frac{20\theta_{if} - 20\theta_{i0} - (8\dot{\theta}_{if} + 12\dot{\theta}_{i0})t_f - (3\ddot{\theta}_{i0} - \ddot{\theta}_{if})t_f^2}{2t_f^3} \\ a_{i4} = \frac{30\theta_{i0} - 30\theta_{if} + (14\dot{\theta}_{if} + 16\dot{\theta}_{i0})t_f + (3\ddot{\theta}_{i0} - 2\ddot{\theta}_{if})t_f^2}{2t_f^4} \\ a_{i5} = \frac{12\theta_{if} - 12\theta_{i0} - (6\dot{\theta}_{if} + 6\dot{\theta}_{i0})t_f - (\ddot{\theta}_{i0} - \ddot{\theta}_{if})t_f^2}{2t_f^5} \end{cases} \quad (13)$$

2.3 Modified Denavit–Hartenberg parameters

In 1955, Denavit and Hartenberg proposed a method for systematically describing multi-axis robots, which are now widely used in robotics research and are known as DH parameters (Denavit et al. 1955). The specific definition model of DH parameters is shown in Fig. 5. Z_{i-1} -axis means the rotation axis of the i th joint, and X_{i-1} -axis means the common perpendicular between the joint i and joint $i + 1$. Y_{i-1} is determined by the right-hand rule following Z_{i-1} -axis and X_{i-1} -axis.

Fig. 5 Standard DH parameters

Besides the DH parameters, there is another definition used widely, which is called modified DH parameters, showing in Fig. 6 (Craig 2009). The difference between these two methods is that the coordinate system is based on the previous joint rather than the next one in standard DH, which means that the frame O_{i-1} is put on joint $i - 1$ not joint i .

2.4 Singularities identification

To identify the singularities, In Robotics, Jacobian matrix is a map that reflects the relationship between joint speed and end-effector speed, shown as Eq. (13):

$$V_e = J[q] \cdot \dot{q} \quad (14)$$

V_e is the velocity of the end-effector, $J[q]$ is the Jacobian matrix, and $\dot{q}$ is the velocity of the joints. To be more specific, V_e includes the linear velocity p_e and angular velocity W_e , the same as $J[q]$:

$$V_e = \begin{bmatrix} p_e \\ W_e \end{bmatrix} = \begin{bmatrix} J_p[q] \\ J_o[q] \end{bmatrix} \cdot \dot{q}. \quad (15)$$

The number of rows in the matrix is the freedom of robot manipulator in operating space, and the number of columns equals the number of joints, so the Jacobian matrix could be divided as

$$\begin{bmatrix} p_e \\ W_e \end{bmatrix} = \begin{bmatrix} J_{P1} J_{P2} J_{P3} J_{P4} J_{P5} J_{P6} \\ J_{W1} J_{W2} J_{W3} J_{W4} J_{W5} J_{W6} \end{bmatrix} \cdot [\dot{q}_1 \dot{q}_2 \dot{q}_3 \dot{q}_4 \dot{q}_5 \dot{q}_6]^T. \quad (16)$$

From above, the p_e and W_e could be mapped from the linear velocity of the joints.

$$\begin{aligned} p_e &= J_{P1} \cdot \dot{q}_1 + J_{P2} \cdot \dot{q}_2 + \dots + J_{P6} \cdot \dot{q}_6 \\ W_e &= J_{W1} \cdot \dot{q}_1 + J_{W2} \cdot \dot{q}_2 + \dots + J_{W6} \cdot \dot{q}_6. \end{aligned} \quad (17)$$

Fig. 6 Modified DH parameters

When the end-effector approaches the kinematic singularities area, the rank of the matrix would decrease, which could conclude that $\det(j[q]) = 0$. From Eq. (18), when $|J[q]|$ approaches 0, q could be so large to damage the robot.

$$q = J[q]^{-1} \cdot V_e = \frac{(J[q])^*}{|J[q]|} \cdot V_e. \quad (18)$$

So $|J[q]| = 0$ is the analyzing condition of the singularity zone of the robot manipulator.

3 Experiment setup

3.1 Robot model

The robot manipulator used in the case study is KUKA KR90. The DH parameters of the robot are shown in Table 3. Based on the DH parameter in Table 3, the robot link model could be built in MATLAB2018a using Robotics Toolbox 10.2, shown in Fig. 7.

Fig. 7 KUKA-KR90 link model

Table 3 DH parameters of the robot

Link i	θ_i (°)	α_{i-1} (°)	a_{i-1} (m)	d_i (m)
1	0	0	0	0
2	0	90	0.350	0
3	0	0	1.150	0
4	0	90	0.041	1.200
5	0	-90	0	0
6	0	-90	0	0

3.2 Data conversion

In this study, one custom wood joint (Fig. 4b) is selected as the research object, with 13 cutting surfaces. The coordinates of 52 vertices of these 13 cutting surfaces are converted from Rhino (Fig. 8) to Matlab (Fig. 9) to construct the via-points of the geometric path, based on which the trajectory would be interpolated using the fifth-order polynomial function. According to the motion control law of the chainsaw, the end-effector cuts the timber along the cutting surfaces. A total of 66 cutting trajectories are formed: (1) trajectories within the 13 cutting surfaces; (2) trajectories from n th surface to $(n + 1)$ th surface; (3) two additional trajectories adding the start and end points.

3.3 Kinematics constraints

The vertices in form of o-xyz coordinate would be transformed into the points in joint space from operating space using inverse kinematics. The results of the 6 joints in form of angular position, velocity and acceleration are shown in Fig. 10.

In Fig. 10, there are 66 curves representing each of the six joints constructed by 66 fifth-order polynomial functions. The coefficients of the functions could be calculated according to Eq. (18) and the constraints for each joint along every trajectory would be obtained. The overall velocity, acceleration and jerk constraints for the objective function T in Eq. (4) can be obtained by setting as:

Fig. 9 Trajectory in Matlab

$$\begin{cases} VC_i = \max \{ \max \{ p_{ij} \} \} \\ WC_i = \max \{ \max \{ \dot{p}_{ij} \} \} \\ JC_i = \max \{ \max \{ \ddot{p}_{ij} \} \} \\ i = \text{number of joint} \\ j = \text{number of trajectory.} \end{cases} \quad (19)$$

3.4 Working space

The work-space of a robot manipulator represents a set of positions that the end-effector could approach, showing the working range of its accessibility. The working space of the manipulator could be expressed as

Fig. 8 Trajectory in Rhino

Fig. 10 Full trajectory

Table 4 Angle range of every joint

Joint	Angle range (°)
1	− 185 to 185
2	− 140 to − 5
3	− 120 to 155
4	− 350 to 350
5	− 125 to 125
6	− 350 to 350

$$W = \begin{cases} p_x = p_x(\theta_i) \\ p_y = p_y(\theta_i) \\ p_z = p_z(\theta_i) \end{cases}, \quad \theta_{i\min} \leq \theta_i \leq \theta_{i\max} \quad (20)$$

Monte Carlo method is one discretization method widely used to compute the work space for complex robots (Rastegar and Fardanesh 1990; Cao et al. 2011). Monte Carlo method gets the approximate work-space by setting the random joint angle of every joints to get the random positions of the end-effector. The angle range of every joint is shown in Table 4.

4 Running the optimization

4.1 Particle swarm optimization for path minimization

Particle swarm optimization is one typical algorithm in swarm intelligence. Eberhard and Kennedy proposed this optimization technique by observing the social behavior of bird flocking (Kennedy and Eberhart 1995). Research has shown the effectiveness of particle swarm optimization in path planning to achieve collision-free trajectories (Hereford 2006; Doctor et al. 2004). Specifically, a group of random particles (random solution) are initialized and the best solution are found through iteration. Within each iteration, the particles update itself by tracking two values: personal best and global best. After finding these two optimal values, the particles update their velocity and position using the following equations.

$$v_i = \omega \times v_i + c_1 \times \text{rand}() \times (p\text{Best}_i - x_i) + c_2 \times \text{rand}() \times (g\text{Best} - x_i) \quad (21)$$

$$x_i = x_i + v_i \quad (22)$$

where v_i is the velocity of the i th particle; $\text{rand}()$ is the random number between 0 and 1; x_i is the current position of the particle; c_1 and c_2 are the learning factors, usually equal 2; ω is the inertia weight, which is a positive number, the larger of the number, the better global searching ability and

weaker local searching ability. The common method is to use the linearly decreasing weight strategy to get the dynamic ω .

$$\omega^{(t)} = \frac{(\omega_{\text{ini}} - \omega_{\text{end}})(G_k - g)}{G_k + \omega_{\text{end}}} \quad (23)$$

where ω_{ini} is the initial inertia weight; ω_{end} is the inertia weight of maximum iteration; G_k is the maximum number of iterations. In this study, $\omega_{\text{ini}} = 1$, $\omega_{\text{end}} = 4e^{(-5)}$. The detailed flow chart is shown as below in Figs. 11 and 12

To combine the algorithm with the parameters in this case, a population of $m \times 13$ particles are created: $\text{swarm} = \{x_1^k, x_2^k, \dots, x_m^k\}$, k means the k th iteration. Within the k th iteration, the i th particle is expressed in a 13-dimension vector: $x_i^k = [(x_{i1}^k), (x_{i2}^k), (x_{i3}^k), \dots, (x_{i13}^k)]$, $i = 1, 2, \dots, m$. Every element of the vector stands for the index of the cutting surface and the index of the vertices within every surface, the j th element of the i th particle is: $x_{ij}^k = (s_{in}^k, \text{ver}_{ip}^k)$, $j = 1, 2, \dots, 13$, $n = 1, 2, \dots, 13$, $p = 1, 2, 3, 4$. Figure 13 explains the detailed layer information, from

Fig. 11 Flow chart of PSO

Fig. 12 Flow chart of robot time optimization using AGA

every particle to the swarm within every step of iteration.

4.2 Adaptive genetic algorithm for travel time minimization

To validate the adaptive genetic algorithm in this study, one closed surface was chosen as an example to optimize the travel time along four edges, which means four time-intervals. The time optimization of the manipulator could be concluded as the joint optimization. The objective functions of j th joint is:

$$f(t) = \min(t_{j1} + t_{j2} + t_{j3} + t_{j4}). \quad (24)$$

According to Eq. (10), the trajectories of the manipulator are constructed by the fifth-order polynomial, the general formula for the trajectories could be expressed as:

$$\begin{cases} h_{ij}(t) = a_{ij5}t_i^5 + a_{ij4}t_i^4 + a_{ij3}t_i^3 + a_{ij2}t_i^2 + a_{ij1}t_i + a_{ij0} \\ i = 1, 2, \dots, N \\ j = 1, 2, \dots, n \end{cases} \quad (25)$$

$h_{ij}(t)$ is the fifth-order polynomial function of the i th trajectory of the j th joint. In one specific surface optimization, $N = 4$. The four positions of the six joints are known concluded from the inverse kinematics, expressed as $X_{j1}, X_{j2}, X_{j3}, X_{j4}, X_{j5}$, and $A_{j1}, A_{j5}, V_{j1}, V_{j5}$ are also known as 0. Based on the known conditions, the relationship between

the unknown a_{ij} and the known X_{ji} could be derived from A_j (a 24×24 matrix) and b_j (vector constructed from X_{ji}).

$$b_j = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ X_{j5} \ 0 \ 0 \ X_{j1} \ 0 \ 0 \ X_{j2} \ X_{j3} \ X_{j4}]^T \quad (26)$$

$$a_j = \text{inv}(A) \times b_j \quad (27)$$

$$a_j = [a_{ij5} \ a_{ij4} \ a_{ij3} \ a_{ij2} \ a_{ij1} \ a_{ij0}]^T. \quad (28)$$

After integrating the John Holland genetic algorithm, the improvement of the process has developed a lot (Whitley 1994). Adaptive genetic algorithm is one method that adjusts the probability of crossover p_c and probability of mutation p_m dynamically according to the fitness value. The adjustment of the two operators are shown in Eqs. (29) and (30):

$$p_c = \begin{cases} p_{c1} - \frac{(p_{c1} - p_{c2})(f' - f_{\text{avg}})}{f_{\text{max}} - f_{\text{avg}}}, & f' \geq f_{\text{avg}} \\ p_{c1}, & f' < f_{\text{avg}} \end{cases} \quad (29)$$

$$p_m = \begin{cases} p_{m1} - \frac{(p_{m1} - p_{m2})(f' - f_{\text{avg}})}{f_{\text{max}} - f_{\text{avg}}}, & f \geq f_{\text{avg}} \\ p_{m1}, & f < f_{\text{avg}} \end{cases} \quad (30)$$

where f is the fitness value of the solution; f' is the larger fitness values of the two solutions to cross; f_{max} is the maximum fitness value of the population; f_{avg} is the average fitness value of the population (Srinivas and Patnaik 1994). The parameters of the

Fig. 13 Layers of iterations PSO

algorithm are selected as follows: population size $M = 100$; $P_{c1} = 0.9$, $P_{c2} = 0.4$, $P_{m1} = 0.1$, $P_{m2} = 0.01$. The $M \times 6$ group of initial combinations of $[t_1 \ t_2 \ t_3 \ t_4]$ as the evolution population would be taken to the algorithm as the independent variables and a_j would be the dependent variables to validate the constraints. After all the optimization for all the joints, the optimized time interval should equal the maximum one within one trajectory to satisfy the kinematics constraints, that is:

$$\begin{cases} t_i = \max \{t_{ji}\} \\ i = 1, 2, \dots, n \\ j = 1, 2, \dots, N. \end{cases} \quad (31)$$

5 Results and analysis

5.1 Distance-optimal simulation

After 300 PSO algorithm iterations, the travel distance decreases from 16.1373 to 15.2010 m. The travel distance

within every cutting surface remain the same, the reduction is diminished by the distance between two cutting surfaces. The trajectories from the start point of n th cutting surface to next start point of $(n + 1)$ th cutting surface are compared in Figs. 14 and 15. The effectiveness of the algorithm in decreasing the distance, from 16.1373 to 15.2010 m (Fig. 16), without singularities in new cutting order, is also validated in KUKA|prc.

Fig. 14 Initial surface-to-surface trajectory

Fig. 15 Optimized the surface to surface trajectory

Fig. 16 Convergence of the distance optimization

This optimization process provides a new method in fabricating complicated components, enhancing the degree in automation in robotic fabrication tool path generation instead of setting the cutting order manually. In addition, the shorter distance could save time and energy.

5.2 Time-optimal simulation

As for the decreased in travel distance, the time optimization is operated within the cutting surface. In the optimization process, the trajectories of the six joints are composed of

four fifth polynomial curves, and precision is 0.0001. The results of the first cutting surface are shown in Table 5. From the optimal results, the travel time along the four edges, four points is reduced under the kinematics constraints of joint velocity, and acceleration.

For all the six joints, the initial total time is 54s, the time is shortened at least 33.3% to 36 s. As shown in Fig. 17a–f, the operating time for each joint would converge to the optimized time, and the optimized operating times are stable. Figure 18 shows after the optimization, the curves of the position, velocity and acceleration are smoother compared to the initial ones, which means the smooth transition after every trajectory ends. Furthermore, this leads to energy saving and deviation reduction in motion control.

6 Conclusion

Based on this research's typical timber joint, this study briefly takes the 6 DOF KUKA-KR90 robot as an example, to convert the geometric information from Rhino into the position information in working space. To implement the objectives functions, the modified DH parameter is chosen to build the link model and the trajectories of the joints are interpolated by fifth-order polynomial functions in joint space. Particle swarm optimization and adaptive genetic algorithm are applied to optimize the path distance and travel time interval respectively.

The two optimization methods achieved the optimal objectives, satisfying the kinematics constraints, and showed effectiveness in decreasing the distance and travel time in the complicated trajectories in this case study. PSO technique improves the automation of robotic trajectory planning in complicated joint fabrication through generating a rational cutting order without singularities. At the same time the trajectory optimized by PSO is also shortened compared to the initial one. The trajectories within one cutting surface optimized by AGA results in time saving and smoother robotic paths, improves the working efficiency and stability by reducing the stress on the actuators in real practice. In real on-site fabrication, this control method could increase working efficiency and improve fabrication accuracy.

Table 5 The optimized time

Time interval	Joint <i>i</i> initial time	1	2	3	4	5	6
		Optimized time					
t_1	18	5.3522	2.0261	11.8355	3.4864	5.9286	5.7879
t_2	9	2.5653	2.7381	4.9952	4.1484	1.2748	2.4834
t_3	18	5.499	3.6438	9.8371	4.9294	1.3568	3.2208
t_4	9	2.0274	3.3993	9.3441	2.5918	5.2089	1.5308
Total times (s)	54	15.4439	11.8074	36.0119	15.1561	13.7691	13.0301

Fig. 17 Convergence of AGA

Future work would focus on different combinations of trajectories technique (like fifth-order B-spline, 3-5-3 polynomial interpolation, etc.) and optimal algorithms like grey

wolf optimization (GWO), ant colony optimization (ACO) to choose the best combination to optimize the distance and

Fig. 18 Comparison of the initial trajectory and AGA time-optimal trajectory

(a) Position of 6 Joints for the Initial Trajectory

(b) Position of 6 Joints for the AGA Time-Optimal Trajectory

(c) Velocity of 6 Joints for the Initial Trajectory

(d) Velocity of 6 Joints for the AGA Time-Optimal Trajectory

(e) Acceleration of 6 Joints for the Initial Trajectory

(f) Acceleration of 6 Joints for the AGA Time-Optimal Trajectory

time. Validation would also be tested on industrial timber which have similar properties to validate the algorithm.

Acknowledgements The case in this research was completed in the workshop in ROBIARCH 2018 and the presented research builds on the case findings. Thanks to the team member Lynn Masuda together working on the model in Rhino. In particular, the authors would warmly like to thank: Achilleas Xydis, Florian Fend and Yuko Ishizu for their helpful technical support and the organization of ETH Zurich.

Funding Not applicable.

Availability of data and material Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are

included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abu-Dakka FJ, Valero FJ, Suárez JL, Mata V (2015) A direct approach to solving trajectory planning problems using genetic algorithms with dynamics considerations in complex environments. *Robotica* 33(3):669
- Agustí-Juan I, Müller F, Hack N, Wangler T, Habert G (2017) Potential benefits of digital fabrication for complex structures: environmental assessment of a robotically fabricated concrete wall. *J Clean Prod* 154:330
- Ata AA (2007) Optimal trajectory planning of manipulators: a review. *J Eng Sci Technol* 2(1):32
- Ata AA, Myo TR (2005) Optimal point-to-point trajectory tracking of redundant manipulators using generalized pattern search. *Int J Adv Robot Syst* 2(3):24
- Balkan T (1998) A dynamic programming approach to optimal control of robotic manipulators. *Mech Res Commun* 25(2):225
- Boscariol P, Gasparetto A, Vidoni R (2012) Jerk-continuous trajectories for cyclic tasks. In: International design engineering technical conferences and computers and information in engineering conference, vol 45035. American Society of Mechanical Engineers, pp 1277–1286
- Cao Y, Lu K, Li X, Zang Y (2011) Accurate numerical methods for computing 2d and 3d robot workspace. *Int J Adv Robot Syst* 8(6):76
- Constantinescu D, Croft EA (2000) Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. *J Robot Syst* 17(5):233
- Cook CC, Ho CY (1984) The application of spline functions to trajectory generation for computer-controlled manipulators. *Computing techniques for robots*. Springer, Boston, MA, pp 101–110
- Craig JJ (2009) Introduction to robotics: mechanics and control, 3/E. Pearson Education India
- Davis L (1991) Handbook of genetic algorithms
- Davis L (1987) Genetic algorithms and simulated annealing
- Denavit J, Hartenberg RS, Mooring BW, Tang GR, Whitney DE, Lozinski CA (1955) 54 Kinematic parameter. *J Appl Mech* 77(2):215–221
- Doctor S, Venayagamoorthy GK, Gudise VG (2004) Optimal PSO for collective robotic search applications. In: Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No.04TH8753), vol 2, pp 1390–1395
- Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the sixth international symposium on micro machine and human science. IEEE, pp 39–43
- Eversmann P, Gramazio F, Kohler M (2017) Robotic prefabrication of timber structures: towards automated large-scale spatial assembly. *Constr Robot* 1(1–4):49
- Faludi J, Bayley C, Bhogal S, Iribarne M (2015) Comparing environmental impacts of additive manufacturing vs traditional machining via life-cycle assessment. *Rapid Prototyp J* 21(1):14–33
- Field G, Stepanenko Y (1996) Iterative dynamic programming: An approach to minimum energy trajectory planning for robotic manipulators. In: Proceedings of IEEE international conference on robotics and automation, vol 3. IEEE, Minneapolis, pp 2755–2760
- Ford S, Despeisse M (2016) Additive manufacturing and sustainability: an exploratory study of the advantages and challenges. *J Clean Prod* 137:1573
- Fu R, Ju HH (2011) Time-optimal trajectory planning algorithm based on AGA for manipulator. *Jisuanji Yingyong Yanjiu* 28(9):3275
- Gasparetto A, Zanutto V (2007) A new method for smooth trajectory planning of robot manipulators. *Mech Mach Theory* 42(4):455
- Gasparetto A, Zanutto V (2008) A technique for time-jerk optimal planning of robot trajectories. *Robot Comput Integr Manuf* 24(3):415
- Gasparetto A, Zanutto V (2010) Optimal trajectory planning for industrial robots. *Adv Eng Softw* 41(4):548
- Gasparetto A, Boscariol P, Lanzutti A, Vidoni R (2012) Trajectory planning in robotics. *Math Comput Sci* 6(3):269
- Gasparetto A, Boscariol P, Lanzutti A, Vidoni R (2015) Path planning and trajectory planning algorithms: A general overview. *Motion and operation planning of robotic systems*. Springer, pp 3–27
- Hao Y, Zu W, Zhao Y (2007) Real-time obstacle avoidance method based on polar coordination particle swarm optimization in dynamic environment. In: 2007 2nd IEEE conference on industrial electronics and applications, pp 1612–1617
- Helm V, Ercan S, Gramazio F, Kohler M (2012) Mobile robotic fabrication on construction sites: DimRob. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, pp 4335–4341
- Helwig S, Wanka R (2008) Theoretical analysis of initial particle swarm behavior. In: Parallel problem solving from nature – PPSN X. Lecture notes in computer science, vol 5199. Springer, Berlin, Heidelberg, pp 889–898
- Hereford JM (2006) A distributed particle swarm optimization algorithm for swarm robotic applications. In: 2006 IEEE international conference on evolutionary computation. IEEE, Vancouver, pp 1678–1685
- Hoffmann M, Mühlenhaller M, Helwig S, Wanka R (2011) Discrete particle swarm optimization for TSP: theoretical results and experimental evaluations. In: Adaptive and intelligent systems. Lecture notes in computer science, vol 6943. Springer, Berlin, Heidelberg, pp 416–427
- Huang J, Hu P, Wu K, Zeng M (2018) Optimal time-jerk trajectory planning for industrial robots. *Mech Mach Theory* 121:530
- Jeevamalar J, Ramabalan S (2012) Optimal trajectory planning for autonomous robots - A review. In: IEEE-International conference on advances in engineering, science and management (ICAESM -2012), pp 269–275
- Jeffers M (2016) Autonomous robotic assembly with variable material properties. *Robotic fabrication in architecture, art and design* 2016. Springer, Cham, pp 48–61
- Keating S, Oxman N (2013) Compound fabrication: a multi-functional robotic platform for digital design and fabrication. *Robot Comput Integr Manuf* 29(6):439
- Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol 4. IEEE, pp 1942–1948
- Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: 1997 IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation, vol 5, pp 4104–4108
- Krieg OD, Schwinn T, Menges A, Li JM, Knippers J, Schmitt A, Schwiager V (2015) Biomimetic lightweight timber plate shells: computational integration of robotic fabrication, architectural geometry and structural design. *Advances in architectural geometry* 2014. Springer, Cham, pp 109–125
- Kyriakopoulos KJ, Saridis GN (1988) Minimum jerk path generation. In: Proceedings. 1988 IEEE international conference on robotics and automation. IEEE, pp 364–369
- Labonnote N, Rønnequist A, Manum B, Rütger P (2016) Additive construction: state-of-the-art, challenges and opportunities. *Autom Constr* 72:347

- Liao X, Wang W, Lin Y, Gong C (2010) Time-optimal trajectory planning for a 6R jointed welding robot using adaptive genetic algorithms. In: 2010 International conference on computer, mechatronics, control and electronic engineering, vol 2. IEEE, pp 600–603
- Luh JYS, Lin CS (1981) Optimum path planning for mechanical manipulators. *J Dyn Syst Meas Control* 103(2):142. <https://doi.org/10.1115/1.3139654>
- Luo LP, Yuan C, Yan RJ, Yuan Q, Wu J, Shin KS, Han CS (2015) Trajectory planning for energy minimization of industry robotic manipulators using the Lagrange interpolation method. *Int J Precis Eng Manuf* 16(5):911
- Petrinec K, Kovacic Z (2007) Trajectory planning algorithm based on the continuity of jerk. In: 2007 Mediterranean conference on control & automation, pp 1–5
- Piazzzi A, Visioli A (1998) Global minimum-time trajectory planning of mechanical manipulators using interval analysis. *Int J Control* 71(4):631
- Popovic D, Fast-Berglund Å, Winroth M (2016) Production of customized and standardized single family timber houses—A comparative study on levels of automation. In: 7th Swedish production symposium
- Ramabalan S, Saravanan R, Balamurugan C (2009) Multi-objective dynamic optimal trajectory planning of robot manipulators in the presence of obstacles. *Int J Adv Manuf Technol* 41(5–6):580
- Rastegar J, Fardanesh B (1990) Manipulation workspace analysis using the Monte Carlo method. *Mech Mach Theory* 25(2):233
- Ratiu M, Adriana Prichici M (2017) Industrial robot trajectory optimization- a review. *MATEC web of conferences*, vol 126, p 2005
- Reyes-Sierra M, Coello CC et al (2006) Multi-objective particle swarm optimizers: a survey of the state-of-the-art. *Int J Comput Intell Res* 2(3):287
- Saravanan R, Ramabalan S, Balamurugan C (2008) Evolutionary optimal trajectory planning for industrial robot with payload constraints. *Int J Adv Manuf Technol* 38(11–12):1213
- Sato A, Sato O, Takahashi N, Kono M (2007) Trajectory for saving energy of a direct-drive manipulator in throwing motion. *Artif Life Robot* 11(1):61
- Shen J, Kong M, Zhu Y (2019) Trajectory optimization algorithm based on robot dynamics and convex optimization. In: 2019 IEEE 3rd advanced information management, communicates, electronic and automation control conference (IMCEC). IEEE, pp 1583–1588
- Shi XH, Liang YC, Lee HP, Lu C, Wang Q (2007) Particle swarm optimization-based algorithms for TSP and generalized TSP. *Inf Process Lett* 103(5):169
- Shiller Z (1996) Time-energy optimal control of articulated systems with geometric path constraints. *J Dyn Syst Meas Control* 118(1):139. <https://doi.org/10.1115/1.2801134>
- Siciliano B, Sciavicco L, Villani L, Oriolo G (2010) Robotics: modeling, planning and control. Springer Science & Business Media
- Simon D (1993) The application of neural networks to optimal robot trajectory planning. *Robot Auton Syst* 11(1):23
- Srinivas M, Patnaik LM (1994) Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans Syst Man Cybern* 24(4):656
- Thomas GB, Weir MD, Hass J, Giordano FR, Korkmaz R (2010) Thomas' calculus. Pearson, Boston
- Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inf Process Lett* 85(6):317
- Verschuere D, Demeulenaere B, Swevers J, De Schutter J, Diehl M (2009) Time-optimal path tracking for robots: a convex optimization approach. *IEEE Trans Autom Control* 54(10):2318
- Wang K-P, Huang L, Zhou C-G, Pang W (2003) Particle swarm optimization for traveling salesman problem. In: Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No.03EX693), vol 3, pp 1583–1585
- Whitley D (1994) A genetic algorithm tutorial. *Stat Comp* 4(2):65
- Xu W, Li C, Liang B, Liu Y, Xu Y (2008) The Cartesian path planning of free-floating space robot using particle swarm optimization. *Int J Adv Robot Syst* 5(3):27
- Xu Z, Li S, Chen Q, Hou B (2015) MOPSO based multi-objective trajectory planning for robot manipulators. In: 2015 2nd international conference on information science and control engineering. IEEE, pp 824–828
- Zanotto V, Gasparetto A, Lanzutti A, Boscariol P, Vidoni R (2011) Experimental validation of minimum time-jerk algorithms for industrial robots. *J Intell Robot Syst* 64(2):197
- Zhang Y, Gong DW, Zhang JH (2013) Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing* 103:172
- Zhang J, Meng Q, Feng X, Shen H (2018) A 6-DOF robot-time optimal trajectory planning based on an improved genetic algorithm. *Robot Biomim* 5(1):1
- Zhao Q, Yan S (2005) Collision-free path planning for mobile robots using chaotic particle swarm optimization. In: Wang L, Chen K, Ong YS (eds) Advances in natural computation. ICNC 2005. Lecture notes in computer science, vol 3612. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11539902_77
- Zhu S, Liu S, Wang X, Wang H (2010) Time-optimal and jerk-continuous trajectory planning algorithm for manipulators. *J Mech Eng* 46(3):47

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.