Hindawi Publishing Corporation
Journal of Robotics
Volume 2016, Article ID 9329131, 9 pages
http://dx.doi.org/10.1155/2016/9329131

Research Article

Optimal Trajectory Planning for Glass-Handing Robot Based on Execution Time Acceleration and Jerk

Honggang Duan, Rongmin Zhang, Fei Yu, Jun Gao, and Yuan Chen

School of Mechanical, Electrical & Information Engineering, Shandong University, Weihai 264209, China

Correspondence should be addressed to Yuan Chen; cyzghysy@sdu.edu.cn

Received 2 November 2015; Accepted 13 January 2016

Academic Editor: Tarek M. Sobh

This study describes a trajectory planning method based on execution time, acceleration, and jerk to ensure that a glass-handing robot runs smoothly at execution time. The minimised objective function consists of the weighted sum of the square of the integral of the execution time, the integral of the acceleration, and the integral of the jerk, all of which are obtained through the weighted coefficient method. A three-dimensional kinematics model of the glass-handing robot is then established and nonuniform fifth-order *B*-splines are used to interpolate its path points. The acceleration and jerk are expressed as functions of time through mathematical simulation. Simulation results show that the designed method for robot trajectory planning not only improves the working efficiency of the glass-handing robot but also ensures that it runs smoothly.

1. Introduction

In the process of producing glass, the glass vessel is transferred from an annealing furnace to a conveyor belt for further processing. The transfer of the product in the process is currently accomplished mainly through manual operation, which is characterised by low efficiency, high labour costs, and high working intensity. The glass-handing robot can automatically transfer glass from the annealing furnace to the conveyor belt, but the vibration generated during its high-speed movement greatly influences the quality of the glass. In particular, the vibration can shake the support bracket, which sustains the main part of the robot at the start and the stop phases. When serious vibration arises during transportation, the glass may be easily damaged, which in turn increases its defect rate. Meanwhile, the vibration also accelerates the loss of robotic parts, thereby reducing the service life of the robot. The robot trajectory optimisation method is a key factor to improve the stability and reduce the vibration of the robot. This study explores how to use the robot trajectory optimisation method to plan a reasonable acceleration in the execution time, smoothen the jerk, and ultimately reduce vibration.

At present, many robot trajectory optimisation methods unilaterally reduce the execution time to optimise the objective function [1–4] or unilaterally decrease the energy consumption of the robot [5–7]. These methods can improve only the unilateral performance of the robot but not its comprehensive performance. Since the development of these methods, Verscheure et al. [8] studied trajectory optimisation in combination with time and energy. van Dijk et al. [9] proposed a trajectory optimisation method for industrial robots with a specified path. Saravanan et al. [10] discussed three kinds of trajectory optimisation methods according to the law of momentum of the limit. Simon and Isik [11] used a trigonometric spline function to interpolate the trajectory of the robot and ensure the smoothness of the change in the pulse value. Gasparetto and Zanotto [12] proposed a new type of robot trajectory planning method based on the proportional relationship between the execution time and the integral of the square of the acceleration. Chen and Li [13] combined the effects of acceleration and pulse value on trajectory optimisation and proposed a trajectory optimisation method that optimises the objective function piecewise. Hossain and Ferdous [14] analysed robot path planning based on the bacterial foraging algorithm.

The above review of the literature shows that current research on trajectory optimisation remains focused on reducing the execution time and energy consumption through kinematics and dynamics analysis or on combining the execution time and acceleration to reduce the pulse value. Decreasing both the execution time and vibration through these methods is difficult. In many practical engineering problems, based on the execution time, planning acceleration and jerk can keep the machine running smoothly and reduce its processing time rationally. For example, the cutter of the numerical control lathe can run smoothly during the acceleration and deceleration of the operation stages and also has a shorter execution time and a higher turning efficiency. A few scholars have conducted an intensive study that combines execution time, acceleration, and jerk to optimise the trajectory. The process must consider acceleration and jerk at different stages of the execution time according to the factors that significantly influence production to reasonably distribute and control the execution time, regulate the acceleration, and guarantee that the robot runs smoothly. For example, a robot significantly accelerating midway during the running time can add more execution time for a smooth change of the jerk at the start and end stages. Most studies use a low-order B-spline curve to interpolate the motion curve, which makes the jerk curve discontinuous and incapable of being optimised piecewise. In this study, the use of the fifth-order B-spline curve and the piecewise optimisation of the acceleration and jerk ensure that the optimisation results of the objective function at different stages become more obvious.

This study uses the glass-handing robot as research object. Firstly, the study establishes a kinematic equation for the robot and then calculates the interpolation points by substituting the path points into the kinematics equation so that the corresponding motion control points can be obtained by back-calculating the interpolation points. Secondly, the B-spline curve is used to simulate the interpolation points, and the MATLAB simulation comparison chart is drawn. Finally, the feasibility of the proposed trajectory optimisation algorithm is verified through the analysis of the MATLAB simulation diagrams.

2. Optimal Trajectory Planning Based on Execution Time, Acceleration, and Jerk

The execution time, acceleration, and jerk of a robot constitute a contradiction. Any change to one part affects the other two parts. For example, increasing the acceleration reduces the execution time and improves work efficiency but also increases the jerk and vibration of the robot. Similarly, increasing the execution time can reduce the acceleration and jerk, which can make the robot run smoothly but reduce its efficiency. Moreover, an increase in jerk increases the acceleration and reduces the execution time, which can improve efficiency but can generate large vibrations. Balancing these three factors is the key to the optimisation of the trajectory of a robot.

According to the differences in the influence of the vibration of the mechanical arm on the glass at varying time periods, this study discusses the no-load and load operations (i.e., the end of the suction cup draws the glass) of a glass-handing robot to reasonably plan the execution time, acceleration, and jerk and therefore solve the contradiction among the three factors. The glass-handing robot tends to generate large vibrations at the start and stop phases, which not only reduces its service life but also damages the glass. Therefore, reducing jerk in these two phases is the key to trajectory optimisation. In addition, the execution time should be increased, and the acceleration should be decreased to avoid damage to the glass caused by large vibrations in the load operation stage and to ensure that the robot can run smoothly. Under the condition of the total execution time, the acceleration at the start and stop phases should therefore be reduced to increase the execution time and reduce the vibration, and the acceleration of the load operation should be reduced to make the robot run smoothly. This process constitutes the basic idea of optimisation of this study, which can be described as follows:

$$I = \min \left\{ A \int_0^{t_p} \left| \frac{\mathrm{d}s^2(t)}{\mathrm{d}t^2} \right| \mathrm{d}t + B \int_0^{t_f} t^2 \mathrm{d}t + [1 - (A + B)] \int_{t_m}^{t_n} \left| \frac{\mathrm{d}^3 s(t)}{\mathrm{d}t^3} \right| \mathrm{d}t \right\}. \tag{1}$$

The constraint equations are as follows:

$$\begin{aligned} \left| s_i^{(1)}(t) \right| &\le V_{i\max}, \\ \left| s_i^{(2)}(t) \right| &\le A_{i\max}, \\ \left| s_i^{(3)}(t) \right| &\le J_{i\max}, \\ s^{(1)}(t_0) &= V_{t_0}, \\ s^{(1)}(t_f) &= V_{t_f}, \\ s^{(2)}(t_0) &= A_{t_0}, \\ s^{(2)}(t_f) &= A_{t_f}, \end{aligned} \tag{2}$$

where A is the proportion of the acceleration in the total optimisation objective function; B is the proportion of the execution time in the total objective function; i represents each joint of the robot; t_p is the execution time at the acceleration or deceleration stage; t_m is the starting time of the smoothly running stage of the load operation; t_n is the ending time of the smoothly running stage of the load operation; t_f is the total execution time for the handing process; $s_i^{(1)}(t)$ is the velocity of the robot; $s_i^{(2)}(t)$ is the acceleration of the robot; $s_i^{(3)}(t)$ is the jerk of the robot; $V_{i\max}$ is the upper limit of the speed of the robot; $A_{i\max}$ is the upper limit of the acceleration of the robot; $J_{i\max}$ is the upper limit of the jerk of the robot; t_0 is the starting time of the handing process; t_f is the ending time of the handing process; V_{t_0} is the starting speed of the handing process; V_{t_f} is the ending speed of the handing process; A_{t_0} is the starting acceleration of the

handing process; and A_{t_f} is the ending acceleration of the handing process. The B-spline curve interpolation planning function is used to implement the idea of the trajectory planning algorithm. In addition, the continuity of the jerk and the "Runge phenomenon" are considered in using the fifth-order B-splines curve.

2.1. Expression of the Fifth-Order B-Spline Curve That Optimises the Objective Function. The B-spline curve is a good curve for trajectory planning, given its advantages of locality, continuity, controllability, zooming ability, and others [15]. Its locality can limit each section of the curve in the range of the limited control points so that changes in the sections of the curve do not affect one another. The continuity of the curve at the inner and end points is beneficial to the fast calculation of the B-spline basis functions. Controllability and zooming ability can make the curve change regularly and independently in different nodes. Thus, in the robot trajectory planning problem, the B-spline curve can be used to construct the ideal curve with the collected interpolation points, and the variables in the joint space function are expressed as a function of time through the B-spline function's inverse calculation. Given $n-1$ control points $(P_0, P_1, \ldots, P_n)$ and one node vector $(\mathbf{U} = [t_0, t_1, \ldots, t_n])$, the p-order B-spline curve and its k-order derivative can be expressed as follows:

$$
\begin{aligned}
&s(t) \\
&\quad = \sum_{j=0}^{m} P_j \left[\frac{t - t_j}{t_{j+p} - t_j} B_j^{p-1}(t) + \frac{t_{j+p+1} - t}{t_{j+p+1} - t_{j+1}} B_{j+1}^{p-1}(t) \right], \\
&\qquad\qquad p > 0,\ t_{\min} \le t \le t_{\max}, \\
&s^{(k)}(t) = \sum_{j=0}^{m} P_j \frac{p!}{(p-k)!} \sum_{i=0}^{k} a_{k,i} B_{j+i}^{p-1}(t), \\
&\qquad\qquad t_{\min} \le t \le t_{\max}, \\
&a_{k,i} = \frac{a_{k-1,i} - a_{k-1,i-1}}{t_{j+p+i-k+1} - t_{j+i}}, \quad i = 1, \ldots, k, \\
&a_{0,0} = 1,
\end{aligned}
\tag{3}
$$

where $B_j^p(t)$ is the B-spline basis function; $B_j^0 = 1$, $(t_j \le t \le t_{j+1})$; and the left is 0. Substituting (3) into (1), the objective function can be redefined as a B-spline curve, expressed as follows:

$$
\begin{aligned}
I = \min \Bigg\{ & A \int_0^{t_p} \left| \sum_{j=0}^{m} P_j B_j^{p(2)}(t) \right| \mathrm{d}t + B \int_0^{t_f} t^2 \mathrm{d}t \\
& + [1 - (A + B)] \int_{t_m}^{t_n} \left| \sum_{j=0}^{m} P_j B_j^{p(3)}(t) \right| \mathrm{d}t \Bigg\}.
\end{aligned}
\tag{4}
$$

2.2. Inverse Calculation of Fifth-Order B-Spline Basis Functions. The inverse calculation of fifth-order B-spline basis functions involves substituting the node vector into the B-spline curve to find the basis function. The expression for the fifth-order B-spline function in section i is assumed to be

$$
\begin{aligned}
s_i(t) = {} & B_0(t) P_{i-1} + B_1(t) P_i + B_2(t) P_{i+1} + B_3(t) P_{i+2} \\
& + B_4(t) P_{i+3} + B_5(t) P_{i+4},
\end{aligned}
\tag{5}
$$

where t represents time; $B_i(t)$ represents the fifth-order B-spline basis functions; and $P_{i-1}, \ldots, P_{i+4}$ are the control points of the fifth-order B-spline curves in section i. According to the continuity of the B-spline curve, two B-spline curves in adjacent segments must be equal at the interpolation points, which means that $s_i(t)$ and $s_{i+1}(t)$ meet $s_i(1) = s_{i+1}(0)$ at $t = 1$ and $t = 0$, respectively. The first to fourth derivatives of the fifth-order B-spline curves in the interpolation points are continuous; that is,

$$
\begin{aligned}
s_i^{(1)}(1) &= s_{i+1}^{(1)}(0), \\
s_i^{(2)}(1) &= s_{i+1}^{(2)}(0), \\
s_i^{(3)}(1) &= s_{i+1}^{(3)}(0), \\
s_i^{(4)}(1) &= s_{i+1}^{(4)}(0).
\end{aligned}
\tag{6}
$$

Through (5) and (6) and the continuity and standardisation of the B-spline curves, the expression of the basis function can be obtained.

2.3. Inverse Calculation of the Control Points of the Fifth-Order B-Spline. In practical applications, calculating a specific B-spline curve requires the node vector to inversely calculate the control point P_i. This study conducts trajectory planning in joint space, which expresses all the joint variables as a function of time. The expected motion of the robot can be described by these joint functions and their first and second derivatives. Therefore, $s(t_k) = q_k$, $k = 0, \ldots, n$. In calculating the unknown control points P_i, $i = 0, \ldots, m$, an equation group is established:

$$
\begin{aligned}
&\mathbf{q}_k \\
&\quad = \begin{bmatrix} B_0^p(t_k) & B_1^p(t_k) & \cdots & B_{m-1}^p(t_k) & B_m^p(t_k) \end{bmatrix}
\begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{m-1} \\ P_m \end{bmatrix}.
\end{aligned}
\tag{7}
$$

Its boundary conditions are shown as follows:

$$
\begin{aligned}
s^{(1)}(t_0) &= v_0, \\
s^{(1)}(t_n) &= v_n, \\
s^{(2)}(t_0) &= a_0, \\
s^{(2)}(t_n) &= a_n.
\end{aligned}
\tag{8}
$$

The above-mentioned constraints can be further expressed in

$$\mathbf{v}_k = \left[B_0^{p(1)}(u_k) \;\; B_1^{p(1)}(u_k) \;\; \cdots \;\; B_{m-1}^{p(1)}(u_k) \;\; B_m^{p(1)}(u_k)\right] \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{m-1} \\ P_m \end{bmatrix},$$

$$\mathbf{a}_k = \left[B_0^{p(2)}(u_k) \;\; B_1^{p(2)}(u_k) \;\; \cdots \;\; B_{m-1}^{p(2)}(u_k) \;\; B_m^{p(2)}(u_k)\right] \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{m-1} \\ P_m \end{bmatrix}. \tag{9}$$

The fourth derivative of the fifth-order B-spline curve is continuous; thus, the node vector used is $\mathbf{t} = [t_0, \ldots, t_0, t_1, \ldots, t_{n-1}, t_n, \ldots, t_n]$, where the numbers of t_0 and t_n are both $p+1$. For the convenience of calculation, the following are set: $\mathbf{A}_i P_i = \mathbf{c}_i$, $P_i = [p_0, p_1, \ldots, p_{m-1}, p_m]^{\mathrm{T}}$, $i = 1, 2, 3$. Equation (10) contains the expressions of $\mathbf{A}_i$ and $\mathbf{c}_i$. By substituting the node vector into the B-spline curve basis function, the expression of matrix $\mathbf{A}_i$ can be obtained. By substituting the interpolation points and constraints into vector $\mathbf{c}_i$, the expression of $\mathbf{c}_i$ can be listed:

$$\mathbf{A}_i = \begin{bmatrix} B_0^p(t_0) & B_1^p(t_0) & \cdots & B_m^p(t_0) \\ B_0^{p(1)}(t_0) & B_1^{p(1)}(t_0) & \cdots & B_m^{p(1)}(t_0) \\ B_0^{p(2)}(t_0) & B_1^{p(2)}(t_0) & \cdots & B_m^{p(2)}(t_0) \\ B_0^p(t_1) & B_1^p(t_1) & \cdots & B_m^p(t_1) \\ \vdots & \vdots & \cdots & \vdots \\ B_0^p(t_{n-1}) & B_1^p(t_{n-1}) & \cdots & B_m^p(t_{n-1}) \\ B_0^{p(2)}(t_n) & B_1^{p(2)}(t_n) & \cdots & B_m^{p(2)}(t_n) \\ B_0^{p(1)}(t_n) & B_1^{p(1)}(t_n) & \cdots & B_m^{p(1)}(t_n) \\ B_0^p(t_n) & B_1^p(t_n) & \cdots & B_m^p(t_n) \end{bmatrix}, \tag{10}$$

$$\mathbf{c}_i = \begin{bmatrix} q_0 \\ v_0 \\ a_0 \\ q_1 \\ \vdots \\ q_{n-1} \\ a_n \\ v_n \\ q_n \end{bmatrix}.$$

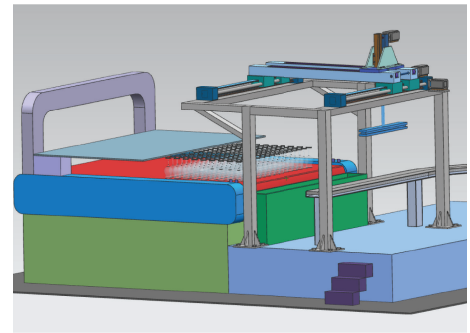

Figure 1: Three-dimensional model of the glass-handing robot.

Given the above analysis and the obtained basis functions, MATLAB can inversely calculate the control point P_i of each mechanical arm of the B-spline. Substituting the node vector into MATLAB can calculate the changes in the acceleration and jerk with time before and after optimisation and can draw the simulation diagram of the said changes.

3. Glass-Handing Robot and Its Kinematics Model

3.1. Three-Dimensional Model of a Glass-Handing Robot and Its Experimental Device. Figure 1 shows the glass-handing robot and its working environment, which is composed of three parts: annealing furnace, glass-handing robot, and conveyor belt. The glass-handing robot in the figure has four degrees of freedom and moves along the X, Y, and Z directions; the suction cup installed in the z-axis can rotate about the z-axis. The glass-handing robot runs from its point of origin to the annealing furnace to draw the glass and then runs to the conveyor belt to put the glass down. However, given that the glass is not placed very regularly, the suction cup needs to be rotated by a certain angle so that the glass can be accurately drawn. Figure 2 shows the experimental device of the glass-handing robot, which is composed of a straight line module moving along the X, Y, and Z directions, a suction cup, and a support bracket. The main advantages of this kind of robot are high reliability, high speed, high bearing capacity, and high precision; this robot is also suitable for work under severe conditions. However, it runs with more vibration and less stability because of the higher rigidity.

3.2. Kinematic Modelling of the Glass-Handing Robot. Constructing the kinematic model of the glass-handing robot facilitates the calculation of the interpolation points of the spline curve, which correspond to the path points of each joint. To build the kinematic model, the robot coordinate system is established, as shown in Figure 3.

The D-H parameters of the glass-handing robot, which are inferred from the coordinates shown in Figure 3, are shown in Table 1.

In the system, T is the base coordinate system, K is the tool coordinate system, and G is the objective coordinate

Figure 2: Experimental device of the glass-handing robot.

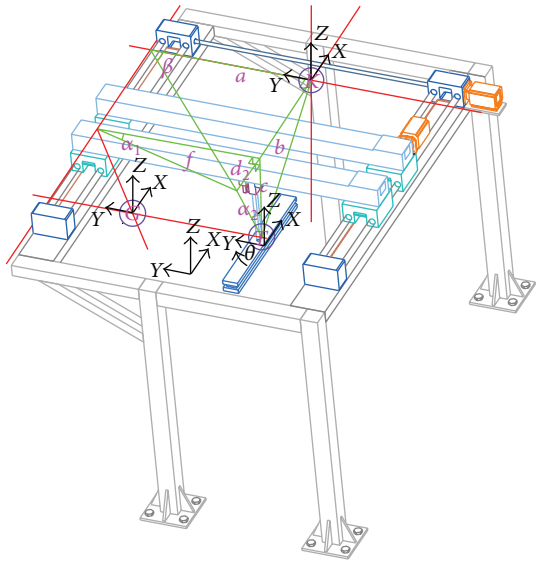

Figure 3: Coordinate system of the glass-handing robot.

Table 1: D-H parameters of the glass-handing robot.

Mechanical arm	$\theta/(^\circ)$	$a_i/(\text{mm})$	$\alpha_i/(^\circ)$	$d_i/(\text{mm})$
X	0	3200	0	0
Y	0	2600	α_1	d_1
Z	0	1400	α_2	0
H	θ	1400	90	d_2

system. The robot runs from the base coordinate system K to the objective coordinate system G to draw the glass and returns to put the glass on the conveyor belt. According to the motion of the robot, four mathematical models of the right triangle are established with the D-H parameters as the data between each coordinate system. According to the established model, the coordinate transformation formula for the position and orientation of the tool coordinate system K relative to the base coordinate system T is as follows:

$$
\begin{aligned}
{}^{K}_{T}\mathbf{T} &= \text{Trans}\,(b,0,0)\,\text{Trans}\,(0,a,0)\,\text{Trans}\,(0,0,c) \\
&\cdot \text{Rot}\,(Z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & b \\ \sin\theta & \cos\theta & 0 & a \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{11}
$$

After obtaining the expression of the position and orientation of the terminal actuator, the unknown parameters in the expression need to be further analysed to complete the calculation. When the distance d_2 between Z and Y mechanical arms and the angle α_2 between them are known, according to the right triangle established in the XZ direction, the elongation in the mechanical arm Z can be obtained as $c = d_2/\tan\alpha_2$. When the distance d_2 between Z and Y mechanical arms and the angle α_1 between Y and X axes are known, according to the right triangle established in the XY direction, the elongation in the mechanical arm Y can be obtained as $f = d_2/\tan\alpha_2$. Thus, combining c and f can obtain the other parameters in Figure 3:

$$
\begin{aligned}
a &= f - d_1 = \frac{d_2}{\tan\alpha_1} - d_1, \\
b &= f \times \tan\beta - d_2 = \frac{d_2\tan\beta}{\tan\alpha_1} - d_2, \\
c &= \frac{d_2}{\tan\alpha_2}.
\end{aligned} \tag{12}
$$

Substituting (12) into the position and orientation of (11) of the terminal actuator, the original equation (11) can be expressed as follows:

$$
{}^{K}_{T}\mathbf{T} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & \dfrac{d_2}{\tan\alpha_1} - d_1 \\ \sin\theta & \cos\theta & 0 & \dfrac{d_2\tan\beta}{\tan\alpha_1} - d_2 \\ 0 & 0 & 1 & \dfrac{d_2}{\tan\alpha_2} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{13}
$$

After obtaining the expression of the position and orientation of the manipulator, the collected parameters should be substituted into this expression to calculate the position points of the manipulator relative to the base coordinate at each moment.

4. Analysis of the Optimisation Results of the Glass-Handing Robot Algorithm

The convergence of the objective function is verified through an iterative method. The implementation steps of the optimisation algorithm are as follows. Firstly, the objective function based on the fifth-order B-spline curve obtained

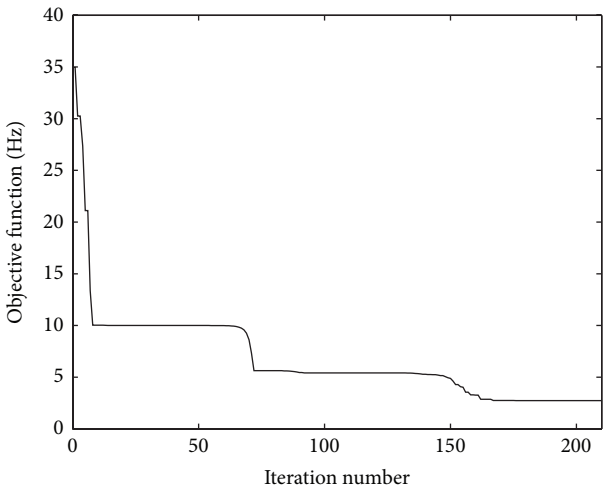

Figure 4: Convergence chart of the objective function.

from the above calculation should be used as input, and the unknown parameters should be set in MATLAB. Secondly, the initial iteration value of the objective function is set as (0, 0, 0) according to the actual situation. Finally, the fminsearch function in the MATLAB toolbox is used to solve the minimum value of the objective function through the iterative method. Figure 4 shows the optimal convergence chart of the objective function.

The limiting conditions of the motion of the glass-handing robot are set according to the requirements of its actual working environment. In (2), $V_{X\max} = 0.5\,\text{m/s}$, $V_{Y\max} = 0.6\,\text{m/s}$, $V_{Z\max} = 0.3\,\text{m/s}$, $A_{X\max} = 0.4\,\text{m/s}^2$, $A_{Y\max} = 0.6\,\text{m/s}^2$, $A_{Z\max} = 0.8\,\text{m/s}^2$, $J_{X\max} = 2\,\text{m/s}^3$, $J_{Y\max} = 1.8\,\text{m/s}^3$, and $J_{Z\max} = 2\,\text{m/s}^3$. In (11), $0^\circ \leq \theta \leq 20^\circ$. Figures 5–7 show the simulation results of the acceleration and jerk of the mechanical arms X, Y, and Z before and after optimisation. The dashed lines represent the simulation results before optimisation, and the solid lines represent the simulation results after optimisation. In clarifying the results to show the advantage of the robot after optimisation, the result calculated by the algorithm in [12] is added for comparison, as expressed by the single line. Its main research on trajectory planning consists of regulating the execution time and reducing the value of jerk. The main function of the mechanical arm X is to adjust the position and orientation of the mechanical arms Y and Z, which do not produce vibration at the robot load stage. Thus, as long as its vibration is controlled under the limit value, it exerts no major influence on the robot. The mechanical arm Y needs to drive the mechanical arm Z when the robot is running, and it runs mainly at the load stage with a long travel. The vibration and frequency produced by the mechanical arm Y greatly influence the robot body and support bracket and glass; thus, reducing the vibration and frequency produced by mechanical arm Y is necessary. The mechanical arm Z is fixed with a suction cup at its bottom and must be positioned such that it directly touches the glass. The mechanical arm Z greatly influences the glass at the start and stop stages, and large vibrations cause the drop and damage of the glass. Thus, the value of the vibration of the start and stop stages should be as small as possible.

Figure 5 shows the simulation results of the acceleration and jerk of the mechanical arm X before and after optimisation and includes the result calculated with the algorithm in [12]. According to Figure 5(a), the optimised acceleration curve of the mechanical arm X is smoother than the acceleration curve before optimisation and that from [12]. This is also reflected in the simulation results of the jerk (Figure 5(b)), which means that the optimised jerk is smaller than the jerk before optimisation and that in [12]. These smaller jerks are conducive to the smooth running of the X arm of the robot and reduce the impact on the whole robot. In addition, Figure 5(a) shows that, before and after the optimisation method used in this study, the change rates of the acceleration of the X axis at the start phase ($0\,\text{s} \leq t \leq 0.5\,\text{s}$) and stop phase ($3.5\,\text{s} \leq t \leq 4\,\text{s}$) decrease gradually. The result in [12] changes more smoothly than the result before optimisation, but using the method of this study, after optimisation, as reflected in Figure 5(b). This means that, in these two phases, the values of the jerk after optimisation are reduced and changes more smoothly than the jerk before optimisation and that from [12]. Moreover, the overall average value of the optimised acceleration is increased, which also ensures that the robot can accomplish the task at the execution time.

Figure 6 shows the simulation results of the acceleration and jerk of the mechanical arm Y before and after optimisation and includes the result calculated with the algorithm in [12]. According to Figure 6(a), in the no-load operation stage ($1\,\text{s} \leq t \leq 5\,\text{s}$), the value and change rate of the acceleration after optimisation are larger than those before optimisation and those from [12], as reflected in Figure 6(b). This finding means that the optimised jerk value is larger than the values before optimisation and those in [12]. This also means that the robot has a longer execution time for mechanical arm Y in the load operation stage. In addition, the peak of the jerk increases but also becomes less than the limit value, and given that the robot is in the no-load operation stage, the vibration generated does not affect the glass. In addition, Figure 6(b) shows that the mechanical arm Y gains a longer execution time in the start phase ($0\,\text{s} \leq t \leq 1\,\text{s}$) and stop phase ($10\,\text{s} \leq t \leq 11\,\text{s}$), and its acceleration value and change rate decrease, as reflected in Figure 6(b) after optimisation. This finding means that the value of the jerk becomes smaller in the start and stop phases. The arm avoids the great impact on the robot body and the glass when the robot suddenly starts and stops. However, the result in [12] does not achieve this effect. In addition, in Figure 6(a), in the load operation stage ($6\,\text{s} \leq t \leq 10\,\text{s}$), the value and change rate of the optimised acceleration become smaller because they have a longer execution time, as reflected in Figure 6(b). This finding means that the optimised jerk changes smoothly, and the value becomes smaller in the load operation stage. Moreover, the maximum peak is less than $1.5\,\text{m/s}^3$. Thus, under the premise of completing the glass movement at the execution

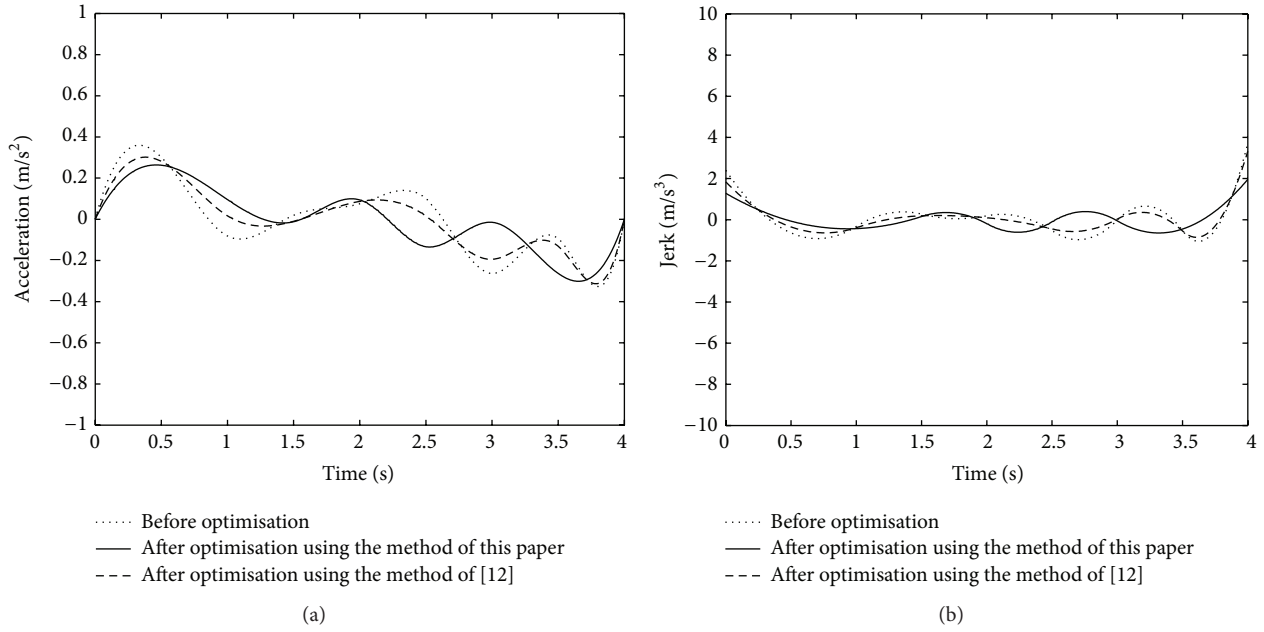

Figure 5: Simulation results of the X joint.

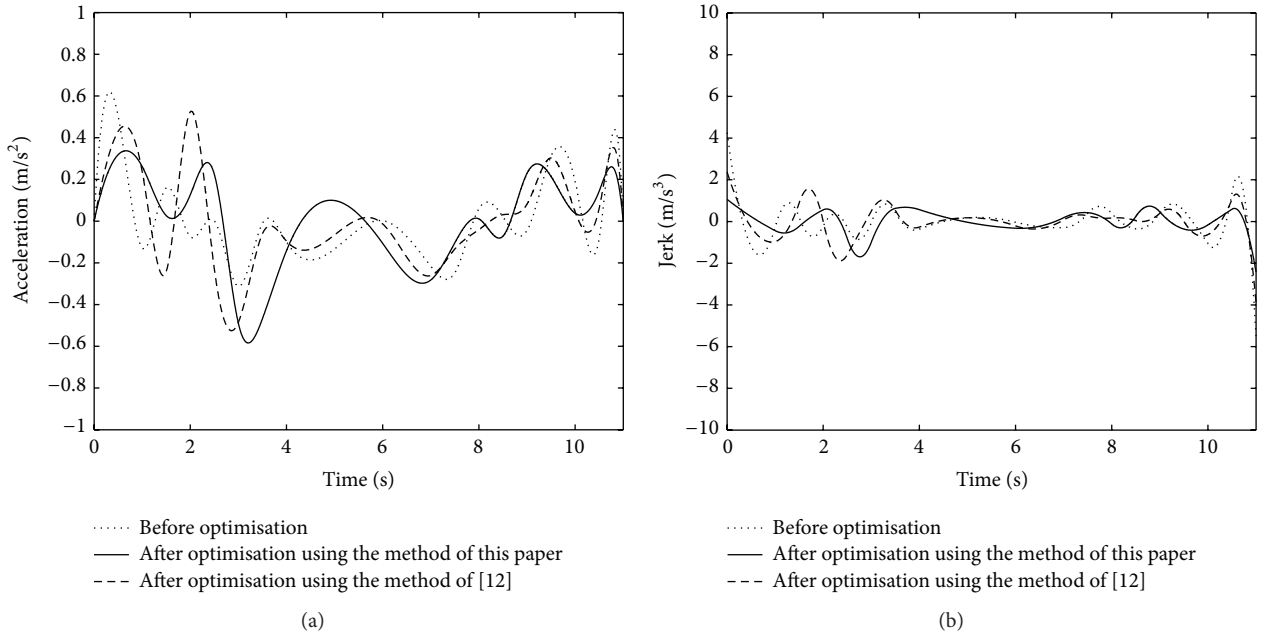

Figure 6: Simulation results of the Y joint.

time, it ensures that the mechanical arm Y can run smoothly and can generate smaller vibrations, thereby protecting more effectively the glass, support bracket, and robot body.

Figure 7 shows the simulation results for the mechanical arm Z before and after optimisation and also includes the result calculated with the algorithm of [12]. In Figure 7(a), the optimised acceleration of the mechanical arm Z is smaller than the acceleration before optimisation and that in [12], and the change rate of the acceleration is significantly smaller, especially in the load operation stage ($3\,\text{s} \leq t \leq 5.5\,\text{s}$). As reflected in Figure 7(b), the optimised jerk becomes smaller and changes smoothly in the load operation stage. In addition, Figure 7(a) shows that the value and change rate of the optimised acceleration decrease in the start phase ($0\,\text{s} \leq t \leq 0.7\,\text{s}$) and stop phase ($5.4\,\text{s} \leq t \leq 6\,\text{s}$). Figure 7(b) shows that the value of the optimised jerk becomes smaller in these two phases, which guarantees that the suction cup fixed in the z-axis does not generate a larger vibration, which in turn damages the glass when the robot picks it up and places it back.

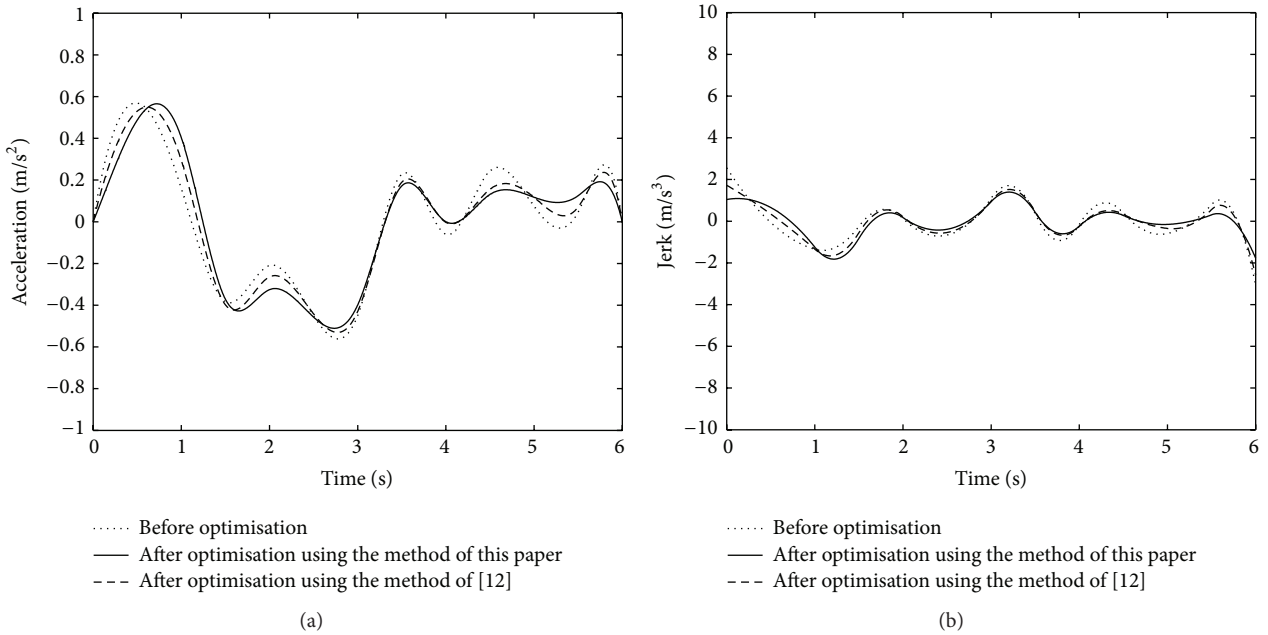

Figure 7: Simulation results of Z joint.

5. Conclusion

This study focuses on the analysis of the simulation results of the acceleration and jerk of a glass-handing robot before and after trajectory optimisation. It also compares the result calculated with Gasparetto's algorithm with the current findings. The analysis method of this experiment involves minimising the objective function, which consists of the integral of the square of the execution time, integral of the acceleration, and integral of the jerk as the main objects under consideration, and achieves the purpose of trajectory optimisation by coordinating the relationship among these three factors.

This study takes the glass-handing robot as an experimental device to analyse the simulation results before and after trajectory optimisation and therefore verifies the feasibility of the algorithm. The desired effect is obtained. Directions for future work include pursuing experiments that further verify the practicability of the objective function, exploring practical applications involving the objective function of the glass packing and stacking robot in the laboratory, and combining the robot with a visual sensor to make it more flexible, efficient, and stable.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Grateful acknowledgement is given to the financial supports from the National Natural Science Foundation of China with Grant no. 51375264 and Science and Technology Major Project of Shandong Province with Grant no. 2015JMRH0218. This work was also partially supported by State Key Laboratory of Robotics and System (HIT) with Grant no. SKLRS-2015-MS-06, China Postdoctoral Science Foundation with Grant nos. 2014T70632 and 2013M530318, and Research Awards Fund for Excellent Young and Middle-aged Scientists of Shandong Province with Grant no. BS2013ZZ008.

References

[1] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000.

[2] C.-S. Lin, P.-R. Chang, and J. Y. S. Luh, "Formulation and optimization of cubic polynomial joint trajectory for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, no. 12, pp. 1066–1073, 1983.

[3] V. Pateloup, E. Duc, and P. Ray, "Corner optimization for pocket machining," *International Journal of Machine Tools and Manufacture*, vol. 44, no. 12-13, pp. 1343–1353, 2004.

[4] A. Piazzi and A. Visioli, "Global minimum-time trajectory planning of mechanical manipulators using interval analysis," *International Journal of Control*, vol. 71, no. 4, pp. 631–652, 1998.

[5] H. J. Kim and B. K. Kim, "Minimum-energy trajectory planning on a tangent for battery-powered three-wheeled omnidirectional mobile robots," in *Proceedings of the International Conference on Control Automation and Systems (ICCAS '10)*, pp. 1701–1706, Gyeonggi-do, Republic of Korea, October 2010.

[6] G. Field and Y. Stepanenko, "Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2755–2760, IEEE, Minneapolis, Minn, USA, April 1996.

[7] V. Nabat, S. Krut, O. Company, P. Poignet, and F. Pierrot, "On the design of a fast parallel robot based on its dynamic model," in *Experimental Robotics*, vol. 39 of *Springer Tracts in Advanced Robotics*, pp. 409–419, Springer, Berlin, Germany, 2008.

[8] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-energy optimal path tracking for robots: a numerically efficient optimization approach," in *Proceedings of the 10th International Workshop on Advanced Motion Control (AMC '08)*, pp. 727–732, Trento, Italy, March 2008.

[9] N. J. M. van Dijk, N. van de Wouw, H. Nijmeijer, and W. C. M. Pancras, "Path-constrained motion planning for robotics based on kinematics constraints," in *Proceedings of the ASME International Design Engineering Technical Conference and the Computer and Information in Engineering Conference (IDETC/CIE '07)*, pp. 1071–1080, Las Vegas, Nev, USA, September 2007.

[10] R. Saravanan, S. Ramabalan, and C. Balamurugan, "Evolutionary optimal trajectory planning for industrial robot with payload constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 11-12, pp. 1213–1226, 2008.

[11] D. Simon and C. Isik, "A trigonometric trajectory generator for robotic arms," *International Journal of Control*, vol. 57, no. 3, pp. 505–517, 1993.

[12] A. Gasparetto and V. Zanotto, "Optimal trajectory planning for industrial robots," *Advances in Engineering Software*, vol. 41, no. 4, pp. 548–556, 2010.

[13] Y. Chen and B. Li, "A piecewise acceleration-optimal and smooth-jerk trajectory planning method for robot manipulator along a predefined path," *International Journal of Advanced Robotic Systems*, vol. 8, no. 4, pp. 184–193, 2011.

[14] M. A. Hossain and I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," *Robotics and Autonomous Systems*, vol. 64, pp. 137–141, 2015.

[15] C. Rossi and S. Savino, "Robot trajectory planning by assigning positions and tangential velocities," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 1, pp. 139–156, 2013.