Contents lists available at SciVerse ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints

Huashan Liu [a,*], Xiaobo Lai [b], Wenxiang Wu [c]

[a] College of Information Science and Technology, Donghua University, Shanghai 201620, China
[b] College of Information Technology, Zhejiang Chinese Medical University, Hangzhou 310053,China
[c] State Key Laboratory of Fluid Power Transmission and Control, Zhejiang University, Hangzhou 310027, China

ARTICLE INFO

Article history:
Received 8 August 2011
Received in revised form
12 August 2012
Accepted 30 August 2012
Available online 2 October 2012

Keywords:
Robot
Trajectory planning
Spline
B-spline
Time optimization

ABSTRACT

In this paper a high smooth trajectory planning method is presented to improve the practical performance of tracking control for robot manipulators. The strategy is designed as a combination of the planning with multi-degree splines in Cartesian space and multi-degree B-splines in joint space. Following implementation, under the premise of precisely passing the via-points required, the cubic spline is used in Cartesian space planning to make either the velocities or the accelerations at the initial and ending moments controllable for the end effector. While the septuple B-spline is applied in joint space planning to make the velocities, accelerations and jerks bounded and continuous, with the initial and ending values of them configurable. In the meantime, minimum-time optimization problem is also discussed. Experimental results show that, the proposed approach is an effective solution to trajectory planning, with ensuring a both smooth and efficiency tracking performance with fluent movement for the robot manipulators.

1. Introduction

As known, the trajectory planning in Cartesian space (operating space or task space) [1–4] is intuitive and easy to observe the motion trail and attitude of the end effector of the robot, especially when there are obstacles to avoid in the operating space. However, such method usually fails to sidestep the problems caused by kinematic singularities. The second method is to carry out the trajectory planning in joint space [5–8], it provides an approach to non-singularity workspace for the robot manipulators, the joint trajectories can be obtained by means of interpolating functions which meet the imposed kinematic and dynamic constraints [9]. Also, it ensures that the end effector of the robot passes through the via-points and would be easier to adjust the trajectory for the system controller in contrast to the former method, but do not guarantee the definite path due to the non-linear relationship between the trajectories in Cartesian space and those in joint space [10,11].

With a review of the most representative interpolating functions in the trajectory planning, polynomials are widely adopted [12]. Generally, when higher accuracy is required for the interpolation, higher degree polynomials will be applied, which probably causes Runge's phenomenon and unstableness of convergence [13]. To eliminate this negative factor, piecewise polynomials are involved in many practical situations. Piecewise line, as a special function of this kind, gains a more satisfactory convergence property, however, it results in non-differentiable points at some internal knots. Therefore, the piecewise polynomials should be at least twice differentiable to guarantee the continuity at every internal knot. Splines, as a class of special functions defined piecewise by multi-order polynomials, are popular curves in tackling interpolating problems with the simplicity of construction, accuracy of evaluation and capacity to approximate complex shapes [14–16]. They are often preferred to polynomials because it yields similar results, even when using low-degree polynomials, while avoiding Runge's phenomenon for higher degrees.

Actually, the motion control system of the robot manipulators acts on the joints, so the smoothness of the joint trajectories would be more important than that of the Cartesian trajectory of the end effector. Aiming to create smooth enough joint trajectories passing through all the internal knots, some typical works with applying different kinds of curves in the planning algorithm are contributed, but most of them fail to obtain satisfactory local support property (if one knot of the curve changes, it only effects the local trajectory besides the knot, without re-computing the entire trajectory) of the trajectories [15–20]. B-splines, characterized by good local support, are invoked in some literatures to obtain both local support property and satisfactory smoothness of the trajectories. Specially, Thompson and Patel [21] developed a planning method for constructing joint trajectories by using B-splines, but it aimed at approximating rather than interpolating

* Corresponding author. Fax: +86 21 67792353.
E-mail address: watson683@163.com (H. Liu).

0736-5845/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.rcim.2012.08.002

the desired sequences of the discrete joint trajectories. Saravanan, Ramabalan, and Balamurugan [22] presented an evolutional theory based methodology for optimal trajectory planning using uniform cubic B-splines, where the robot joint accelerations and jerks of the resulting trajectories can be restricted within their limiting values. Gasparetto and Zanotto [9] applied quintic B-spline in the interpolation to generate smooth joint trajectories, the proposed method enables one to impose kinematic constraints, expressed as upper bounds on the absolute values of velocity, acceleration and jerk of the robot joint. By the same authors [19,20], an objective function composed of two terms (one is proportional to the execution time and the other is proportional to the integral of the squared jerk) is minimized during the cubic splines and quintic B-splines based planning to get the optimal trajectory. It's worth mentioning that, to consider the kinematic constraints in the on-line trajectory generation problem [23], Kröger discussed cases of constant [24] and non-constant [25] kinematic motion constraints imposed on the robots in depth.

High jerk of the robot joint can heavily excite the resonance frequencies of the body structure, creating vibrations, and slow down the tracking speed, as well as affect the tracking precision, so keeping the absolute value of jerk in a relative small bounded area is vital. Furthermore, if the continuity of the jerk is guaranteed, the flexible impact created by the joint actuator will be small. With respect to the smoothing techniques that could be found in previous related studies, the method described in this work ensures that the trajectory in Cartesian space is twice continuous differentiable, while in the joint space, the velocity, acceleration and jerk are all continuous and bounded. Moreover, the initial and the ending value of the velocity, acceleration and jerk of each robot joint can be configured almost arbitrarily as needed. Moreover, considering both smooth performance and efficiency execution, optimization for minimum-time trajectory tracking is also presented.

The rest of this paper is organized as follows. Section 2 details the trajectory planning method in Cartesian space by splines. In Section 3, the general formula of multi-degree B-splines is discussed, based on it, the trajectory planning in joint space is presented. In Section 4, minimum-time optimization is presented. Then, experimental results on a PUMA 560 structured 6 DOF serial robot with revolute joints are shown in Section 5. Finally, some conclusions are given in Section 6.

2. Trajectory planning in Cartesian space

Given the discrete sequence of interpolation samples: $w_i = f(t_i)$, $i=0, 1, \ldots, n$, where $0 \le a = t_0 < t_1 < t_2 < \cdots < t_n = b$. If there exists $s(t)$ with n piecewise polynomials as

$$s(t) = \begin{cases} s_1(t) & t \in [t_0, t_1] \\ s_2(t) & t \in [t_1, t_2] \\ \vdots & \\ s_n(t) & t \in [t_{n-1}, t_n] \end{cases}$$

$$s_j(t) = u_{k,\ j}t^k + u_{k-1,\ j}t^{k-1} \ldots + u_{1,\ j}t + u_{0,j} \quad j = 1,2,\ldots,n \tag{1}$$

where $u_{k,\,j}, u_{k-1,\,j}, \ldots, u_{0,\,j}$ are constant coefficients, k is the degree of the polynomial $s_j(t)$, that satisfies

(i) the interpolating property, $s(t_i) = w_i = f(t_i)$;
(ii) the curves to join up, $s_j(t_j) = s_j + 1(t_j)$;
(iii) $(k-1)$ times continuous differentiable, $s_j'(t_j) = s_{j+1}'(t_j)$, $s_j''(t_j) = s_{j+1}''(t_j)$, $\ldots$, $s_j^{(k-1)}(t_j) = s_{j+1}^{(k-1)}(t_j)$.

Then, $s(t)$ is a spline interpolating function of degree k for the discrete sequence $w_i = f(t_i)$ [26].

In mathematics, a spline is a special function defined piecewise by polynomials. In engineering applications, spline interpolation is often preferred to polynomial interpolation because the interpolation error can be made small even when using low-degree polynomials for the spline. As a positive result of this feature, spline interpolation avoids the problem of Runge's phenomenon which occurs when using high degree polynomials.

Aiming to enhance the smooth movement performance of the robot manipulators, we invoke cubic splines in the interpolation to obtain a twice continuous differentiable trajectory in Cartesian space, i.e., the acceleration along the coordinate x (or y, z) for time t varies continuously. Accordingly, the expression of function $s_j(t)$ can be written as

$$s_j(t) = u_{3,j}t^3 + u_{2,j}t^2 + u_{1,j}t + u_{0,j} \quad j = 1, 2, \ldots, n \tag{2}$$

Noting that $s_j''(t)$is a first-order polynomial on the closed interval $[t_{j-1}, t_j]$, we suppose the value of $s_j''(t)$at the ends of this interval are known: $s''(t_{j-1}) = M_{j-1}, s''(t_j) = M_j$, then

$$s_j''(t) = \frac{(t_j - t)M_{j-1} + (t - t_{j-1})M_j}{h_j} \tag{3}$$

where $h_j = t_j - t_{j-1}$.

By calculating the integrals of (3), we obtain the general expression for the cubic spline at any time t in $[t_{j-1}, t_j]$:

$$s_j(t) = \frac{(t_j - t)^3 M_{j-1} + (t - t_{j-1})^3 M_j + (t_j - t)(6w_{j-1} - M_{j-1}h_j^2) + (t - t_{j-1})(6w_j - M_j h_j^2)}{6h_j} \tag{4}$$

Eq. (4) indicates that there are $(n+1)$ unknown variables $(M_0, M_1, \ldots, M_n)$ for $s(t)$, to get the complete expression of the spline, we need to construct $(n+1)$ independent equations for $M_0, M_1, \ldots, M_n$.

Specially, using condition (iii), we get

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = \gamma_j \quad j = 1,2,\ldots,n-1 \tag{5}$$

where $\mu_j = h_j/(h_j + h_{j+1})$, $\lambda_j = 1 - \mu_j$, $\gamma_j = 6[(w_{j+1} - w_j)/h_{j+1} - (w_j - w_{j-1})/h_j]/(h_j + h_{j+1})$.

For the n cubic polynomials comprising $s(t)$, there are $(n-1)$ interior knots, giving us $(n-1)$ equations in the form of (5). To solve the $(n+1)$ unknown variables, we still require two other constraint conditions, which can be imposed upon the problem for different reasons.

Case 1. $s'(t_0) = w_n'$, $s'(t_n) = w_n'$.

The significance of these two constraint conditions on strategy of motion control in Cartesian space is: we can set the initial and ending value of velocity along the desired trajectory according to the actual needs. Generally, we set $w_0' = w_n' = 0$.

By such restrictions, we get two equations as

$$2M_0 + M_1 = \gamma_0,\ M_{n-1} + 2M_n = \gamma_n \tag{6}$$

where $\gamma_0 = 6[(w_1 - w_0) - h_1 w_0']/h_1^2$, $\gamma_n = 6[h_n w_n' - (w_n - w_{n-1})]/h_n^2$.

From (5) and (6), we can obtain the $(n+1)$ order linear equations as

$$\begin{bmatrix} 2 & 1 & & & & \\ \mu_1 & 2 & \lambda_1 & & & \\ & \mu_2 & 2 & \lambda_2 & & \\ & & \ddots & \ddots & & \\ & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n-1} \\ \gamma_n \end{bmatrix} \tag{7}$$

where μ_i, λ_i and $\gamma_i (i = 0, 1, \ldots, n)$ are the known quantities as referred in (5) and (6), M_i are the unknown variables.

Case 2. $s''(t_0) = w_0''(=M_0)$, $s''(t_n) = w_n''(=M_n)$.

For the purpose of this paper, we set two additional conditions:

$$s''(t_0) = M_0 = 0,\ s''(t_n) = M_n = 0 \tag{8}$$

which results in the natural cubic spline and guarantees the zero initial and ending accelerations as well twice differentiable smoothness along the trajectory.

Finally, with transformations, we can get the linear equations as

$$\begin{bmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n-2} \\ \gamma_{n-1} \end{bmatrix} \tag{9}$$

Generally, the interpolation is implemented in equal time intervals, thus λ_k and μ_k are often set as $\lambda_k = \mu_k = 1/2$.

3. Trajectory planning in joint space

The smoothness of trajectories in joint space, which directly effects the stability of the output torque of actuators, is vital in trajectory planning. Multi-degree B-splines are characterized by good smoothness and local support, thus it is often used in the trajectory planning recently.

By applying inverse kinematics transformation, we can obtain the trajectory $\boldsymbol{p}$ in joint space (time sequence of joint angle) from the trajectory in Cartesian space:

$$\boldsymbol{p} = \{\boldsymbol{p}_i\} = \{(t_i, \theta_i)\} \quad (i = 0,1,\ldots,n) \tag{10}$$

where θ_i denotes the joint angle at time t_i.

To facilitate presentations and unify formula expressions, the definitional domain of B-splines is usually normalized into range of [0,1]. Therefore, by Accumulative Chord Length method, we normalize the time variable t_i to gain the knot variable u_i of k-th-degree B-splines, furthermore, to make the trajectory parameters at the endpoints of trajectory controllable, we extend the knot vector by k additional knots at each end, i.e.

$$u_i = \begin{cases} 0, \quad i = 0, 1, \ldots, k \\ u_{i-1} + \frac{t_{i-k} - t_{i-k-1}}{t_n - t_0}, \quad i = k+1, k+2, \ldots, n+k-1 \\ 1, \quad i = n+k, n+k+1, \ldots, n+2k \end{cases} \tag{11}$$

When using k-th-degree B-splines to interpolate $(n+1)$ points p_i $(i = 0, 1, \ldots, n)$, the trajectory can be expressed as

$$\boldsymbol{p}(u_{i+k}) = \sum_{j=i}^{i+k-1} \boldsymbol{d}_j N_{j,k}(u_{i+k}) = \boldsymbol{p}_i, \quad u_{i+k} \in [u_k, u_{n+k}] \tag{12}$$

where d_j $(j = 0, 1, \ldots, n+k-1)$ is the control points to be solved; $N_{j,k}(u_{i+k})$ is the basis function of the k-th-degree B-splines, which can be gained from the Cox-de Boor recursion formula [27]; u_{i+k} is the knot variable, $i = 0, 1, \ldots, n$.

As in (12), to solve the $(n+k)$ control points, we already have $(n+1)$ equations, there still needs another $(k-1)$ equations, which can be created as kinematic constraint conditions at two endpoints.

In general, we often take the boundary conditions of the r-th order derivatives for B-splines as the additional equations, where r is a natural constant. The r-th order derivative $p^r(u)$ of the B-splines can be obtained by applying the formula below:

$$p^r(u) = \frac{d^r}{du^r} \sum_{j=0}^{n} d_j N_{j,\ k}(u)$$

$$= \sum_{j=i-k+r}^{i} \boldsymbol{d}_j^r N_{j,k-r}(u),\ u \in [u_i, u_{i+1}] \subset [u_k, u_{n+k}] \tag{13}$$

where

$$\boldsymbol{d}_j^l = \begin{cases} \boldsymbol{d}_j \quad l = 0 \\ (k-l+1) \frac{\boldsymbol{d}_j^{l-1} - \boldsymbol{d}_{j-1}^{l-1}}{u_{j+k-l+1} - u_j} \quad \begin{array}{l} l = 1, 2, \cdots, r \\ j = i-k+l, i-k+l+1, \cdots, i \end{array} \end{cases} \tag{14}$$

Obviously, from (13), we can find that r-th order derivatives of k-th-degree B-splines can be expressed as $(k-r)$-th-degree B-splines. When $r = 1$, 2 and 3, the derivatives represent the vectors of the velocity $\boldsymbol{V}(t)$, acceleration $\boldsymbol{A}(t)$ and jerk $\boldsymbol{J}(t)$ of the robot joints, respectively, i.e.

$$\boldsymbol{V}(t) = \boldsymbol{p}'(u) = \sum_{j=i-k+1}^{i} \boldsymbol{d}_j^1 N_{j,k-1}(u) \tag{15}$$

$$\boldsymbol{A}(t) = \boldsymbol{p}''(u) = \sum_{j=i-k+2}^{i} \boldsymbol{d}_j^2 N_{j,k-2}(u) \tag{16}$$

$$\boldsymbol{J}(t) = \boldsymbol{p}'''(u) = \sum_{j=i-k+3}^{i} \boldsymbol{d}_j^3 N_{j,\ k-3}(u) \tag{17}$$

Since the multiplicity at the two endpoints of trajectory is k, the start and the end control points are the corresponding start and the end via points of the trajectory to be planned. Therefore, according to (13), we obtain

$$\boldsymbol{V}(t_0) = \boldsymbol{p}_0' = \boldsymbol{p}'(u_k) = \sum_{j=k-k+1}^{k} \boldsymbol{d}_j^1 N_{j,\ k-1}(u_k) = \boldsymbol{d}_1^1 \tag{18}$$

$$\boldsymbol{V}(t_n) = \boldsymbol{p}_n' = \boldsymbol{p}'(u_{n+k}) = \sum_{j=n+k-1-k+1}^{n+k-1} \boldsymbol{d}_j^1 N_{j,k-1}(u_{n+k}) = \boldsymbol{d}_{n+k-1}^1 \tag{19}$$

$$\boldsymbol{A}(t_0) = \boldsymbol{p}_0'' = \boldsymbol{p}''(u_k) = \sum_{j=k-k+2}^{k} \boldsymbol{d}_j^2 N_{j,\ k-2}(u_k) = \boldsymbol{d}_2^2 \tag{20}$$

$$\boldsymbol{A}(t_n) = \boldsymbol{p}_n'' = \boldsymbol{p}''(u_{n+k}) = \sum_{j=n+k-1-k+2}^{n+k-1} \boldsymbol{d}_j^2 N_{j,\ k-2}(u_{n+k}) = \boldsymbol{d}_{n+k-1}^2 \tag{21}$$

$$\boldsymbol{J}(t_0) = \boldsymbol{p}_0''' = \boldsymbol{p}'''(u_k) = \sum_{j=k-k+3}^{k} \boldsymbol{d}_j^3 N_{j,k-3}(u_k) = \boldsymbol{d}_3^3 \tag{22}$$

$$\boldsymbol{J}(t_n) = \boldsymbol{p}_n''' = \boldsymbol{p}'''(u_{n+k}) = \sum_{j=n+k-1-k+3}^{n+k-1} \boldsymbol{d}_j^3 N_{j,\ k-3}(u_{n+k}) = \boldsymbol{d}_{n+k-1}^3 \tag{23}$$

where $\boldsymbol{V}(t_0)$, $\boldsymbol{V}(t_n)$, $\boldsymbol{A}(t_0)$, $\boldsymbol{A}(t_n)$, $\boldsymbol{J}(t_0)$ and $\boldsymbol{J}(t_n)$ denote the angular velocity, acceleration and jerk at the initial and the ending moment of the trajectory in joint space, respectively; $\boldsymbol{d}_1^1$, $\boldsymbol{d}_{n+k-1}^1$, $\boldsymbol{d}_2^2$, $\boldsymbol{d}_{n+k-1}^2$, $\boldsymbol{d}_3^3$ and $\boldsymbol{d}_{n+k-1}^3$ can be expressed with the control points $\boldsymbol{d}_j$ by applying the recursion formula (14).

Specially, a k-th-degree B-spline curve is always contained in the convex hull of its control points. In general, the lower the degree of the B-splines is, the stronger convex hull property it has, i.e, the closer the B-spline curve follows its control points. Therefore, just to satisfy

$$\max(|d_{jm}^1|) \le \sup|V_m|, j = 1,2, \cdots, n+k \tag{24}$$

$$\max(|d_{jm}^2|) \le \sup|A_m|, j = 2, 3, \cdots, n+k \tag{25}$$

$$\max(|d^3_{jm}|) \le \sup|J_m|, j=3, 4, \cdots, n+k \tag{26}$$

where d^1_{jm}, d^2_{jm} and d^3_{jm} denote the j-th control point of the B-spline velocity, acceleration and jerk curves for the m-th joint; V_m,A_m and J_m denote the velocity, acceleration and jerk of the m-th joint.

Accordingly, when using septuple B-splines ($k=7$) in the joint trajectory interpolation, we can take (18)~(23) as the 6 additional conditions and (24)~(26) as the kinematic constraints, where the velocities, the accelerations and the jerks at the initial and the ending moments can be configured arbitrarily as needed. Usually they are all set to zero to reduce the flexible impact and improve the fluency of the movement for each joint of robot.

Similarly, when using quintic B-splines ($k=5$), (18)~(21) can be taken as the 4 additional conditions and (24)~(25) as the kinematic constraints; when using cubic B-splines ($k=3$), (18)~(19) as the 2 additional conditions and (24) as the kinematic constraint.

4. Time optimization

Time optimization is to solve the minimum execution time problem under given kinematic constraints during the trajectory planning in joint space, the expression in mathematical terms can be written as

$$f(\boldsymbol{x}) = \min\sum_{i=0}^{n-1}\boldsymbol{x}_i \quad i=0,1,\ldots,n-1$$
$$\text{s.t.} \begin{cases} c_m(\boldsymbol{x}) = \max\limits_{j=1,2,\ldots,n+6}(|d^1_{jm}(\boldsymbol{x})|) - \sup|V_m| \le 0 \\ c_{N+m}(\boldsymbol{x}) \max\limits_{j=2,3,\ldots,n+6}(|d^2_{jm}(\boldsymbol{x})|) - \sup|A_m| \le 0 \\ c_{2N+m}(\boldsymbol{x}) \max\limits_{j=3,4,\ldots,n+6}(|d^3_{jm}(\boldsymbol{x})|) - \sup|J_m| \le 0 \end{cases} \tag{27}$$

where $\boldsymbol{x}=[x_1\,x_2\,\ldots\,x_n]^{\mathrm{T}}$,$x_i=\Delta t_i = t_{i+1}-t_i$ and each Δt_i has a lower bound satisfied $\inf|\Delta t_i| \ge \max\limits_{m=1,2,\ldots,N}\left(\frac{p_{(i+1)m}-p_{im}}{\sup|V_m|}\right)$, N is the degree of freedom of the robot manipulator. $c_m(\boldsymbol{x})$, $c_{N+m}(\boldsymbol{x})$, $c_{2N+m}(\boldsymbol{x})$ ($m=1, 2, \ldots, N$) is the difference between the velocity, acceleration, jerk calculated by the spline equation and the given supremum of the velocity, acceleration, jerk for the m-th joint of the robot manipulator, respectively.

To optimize the nonlinear constraints as described by (27), we order

$$\boldsymbol{T}_{\inf} = [\inf|\Delta t_0|\ \inf|\Delta t_2|\ \ldots\ \inf|\Delta t_{n-1}|]^{\mathrm{T}} \tag{28}$$

$$\begin{cases} k_1 = \max\left(\frac{d^1_{jm}}{\sup|V_m|}\right) & j=1,2,\ldots,n+6 \quad m=1,2,\ldots,N \\ k_2 = \max\left(\frac{d^2_{jm}}{\sup|A_m|}\right) & j=2,3,\ldots,n+6 \quad m=1,2,\ldots,N \\ k_3 = \max\left(\frac{d^3_{jm}}{\sup|J_m|}\right) & j=3,4,\ldots,n+6 \quad m=1,2,\ldots,N \end{cases} \tag{29}$$

then, according to k_1, k_2 and k_3, we define the initial value $\boldsymbol{x}_0$ of the knot vector $\boldsymbol{x}$:

$$\boldsymbol{x}_0 = \max(1,k_1,\sqrt{k_2},\sqrt[3]{k_3}) \times \boldsymbol{T}_{\inf} \tag{30}$$

The Sequential Quadratic Programming (SQP) method, which is characterized by super-linear convergence, is adopted to solve (27) with linearizing the nonlinear constraints. Design a Lagrange function:

$$L(\boldsymbol{x}, \gamma) = f(\boldsymbol{x}) - \gamma^{\mathrm{T}}C(\boldsymbol{x}) \tag{31}$$

where $\gamma = [\gamma_1, \gamma_2, \cdots, \gamma_{3N}]^{\mathrm{T}}$is the Lagrange multiplier, $C(\boldsymbol{x}) = [c_1(\boldsymbol{x}), c_2(\boldsymbol{x}), \cdots, c_{3N}(\boldsymbol{x})]^{\mathrm{T}}$. $f(\boldsymbol{x})$ and c_i ($\boldsymbol{x}$) ($i=1, 2, \ldots, 3N$) in $C(\boldsymbol{x})$ are as defined in (27).

If there exists a certain point $\boldsymbol{x}^*$ making the gradient of the Lagrange function $\nabla L(\boldsymbol{x}^*, \gamma)=0$, i.e., $\nabla f(\boldsymbol{x}^*) - \gamma^{\mathrm{T}}\nabla C(\boldsymbol{x}^*)=0$, then $\boldsymbol{x}^*$ is the K-T point, i.e., the solution of (27). By Newton–Raphson method [28], we obtain the k-th sub-problem of the SQP method:

$$\min_{\boldsymbol{d}\in \mathrm{R}^n}\left(\boldsymbol{g}_k{}^{\mathrm{T}}\boldsymbol{d} + \tfrac{1}{2}\boldsymbol{d}^{\mathrm{T}}\mathrm{B}_k\boldsymbol{d}\right)$$
$$s.t.\ \ a_i(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{d} + c_i(\boldsymbol{x}_k) \le 0 \ \ i=1,2,\cdots,3N \tag{32}$$

where $\boldsymbol{g}_k = \mathrm{g}(\boldsymbol{x}_k) = \nabla f(\boldsymbol{x}_k)$, $A(\boldsymbol{x}_k) = [a_1(\boldsymbol{x}_k), a_2(\boldsymbol{x}_k), \cdots, a_{3N}(\boldsymbol{x}_k)] = \nabla C(\boldsymbol{x}_k)^{\mathrm{T}}$, $\boldsymbol{B}_k \in \mathrm{R}^{n\times n}$ is the approximation of the Hessian matrix of the Lagrange function.

Since the solution of the k-th sub-problem of the SQP method is written as $\boldsymbol{d}_k$, then there exists the K-T equation:

$$\boldsymbol{g}_k + \boldsymbol{B}_k\boldsymbol{d}_k = A(\boldsymbol{x}_k)\gamma_k \quad (\gamma_k)_i \ge 0 \ \ i=1,2,\cdots,3N$$
$$\gamma_k^{\mathrm{T}}[C(\boldsymbol{x}_k) + A(\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{d}_k] = 0 \tag{33}$$

We can work out the time optimization problem with the SQP method by following the three steps as below [28,29]:

Step 1: Solve $\boldsymbol{x}_0$ by (30), then set $\sigma>0$, $\delta>0$, $\varepsilon \ge 1$, $k=0$, and calculate the Hessian matrix $\boldsymbol{B}_0$;

Step 2: Solve $\boldsymbol{d}_k$ by (32). If $||\boldsymbol{d}_k|| \le \varepsilon$, then break; if not, work out $\alpha_k = \underset{0\le\alpha\le\delta}{\operatorname{argmin}} P(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k, \sigma) + \varepsilon_k$, where the function $P(\boldsymbol{x},\sigma) = f(\boldsymbol{x}) + \sigma\left[\sum_{i=1}^{3N}|\min(0,\ c_i(\boldsymbol{x}))|\right]$ is the L_1 exact penalty function;

Step 3: Calculate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{d}_k$, then update the Hessian matrix $\boldsymbol{B}_{k+1}$ by Broyden–Fletcher–Goldfarb–Shanno (BFGS) method[29]:

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{q}_k\boldsymbol{q}_k{}^{\mathrm{T}}}{\boldsymbol{q}_k{}^{\mathrm{T}}\boldsymbol{s}_k} - \frac{\boldsymbol{B}_k{}^{\mathrm{T}}\boldsymbol{B}_k}{\boldsymbol{s}_k{}^{\mathrm{T}}\boldsymbol{B}_k\boldsymbol{s}_k},$$

where

$$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k,\ \boldsymbol{q}_k = \nabla f(\boldsymbol{x}_{k+1}) + \sum_{i=1}^{3N}\gamma_i\nabla g_i(\boldsymbol{x}_{k+1}) - \left[\nabla f(\boldsymbol{x}_k) + \sum_{i=1}^{3N}\gamma_i\nabla g_i(\boldsymbol{x}_k)\right].$$

Order $k=k+1$, return to Step 2.

Importantly, the exact penalty function $P(\boldsymbol{x}, \sigma)$ is not always differentiable in practical applications, taking it as the value function may produce Maratos effect[29], which affects the convergence of the algorithm. To deal with this problem, we add a second-order correction step:

Replace the $\boldsymbol{g}_k$ with $\hat{\boldsymbol{g}}_k$, which is written as

$$\hat{\boldsymbol{g}}_k = \boldsymbol{g}_k + \frac{1}{2}\sum_{i=1}^{3N}(\gamma_k)_i[\nabla c_i(\boldsymbol{x}_k) - \nabla c_i(\boldsymbol{x}_k + \boldsymbol{d}_k)] \tag{34}$$

where $\boldsymbol{d}_k$ and γ_k are as defined in (33).

By solving the modified SQP problem, we obtain the corresponding numerical solution $\hat{\boldsymbol{d}}_k$. When $\boldsymbol{x}_k$ is sufficiently close to the optimal solution $\boldsymbol{x}^*$, order $x_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k + \hat{\boldsymbol{d}}_k$in Step 3.

Finally, by following the steps above successively, the time vector $\boldsymbol{x}$ converges to the optimum value $\boldsymbol{x}^*$, then the minimum execution time $f(\boldsymbol{x}^*)$ is obtained.

5. Experimental results

To verify the effectiveness of the proposed strategy, we implement experimental tests on a platform (Fig. 1) of PUMA560 structured 6-DOF serial robot—QJ-I (Fig. 2a), with its D-H link

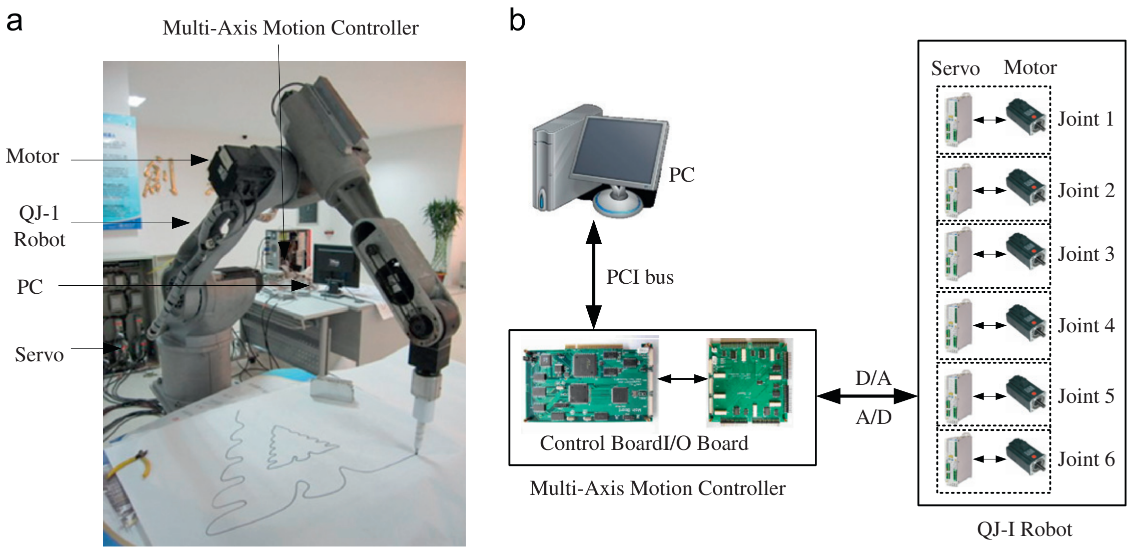


Fig. 1. Experimental platform.

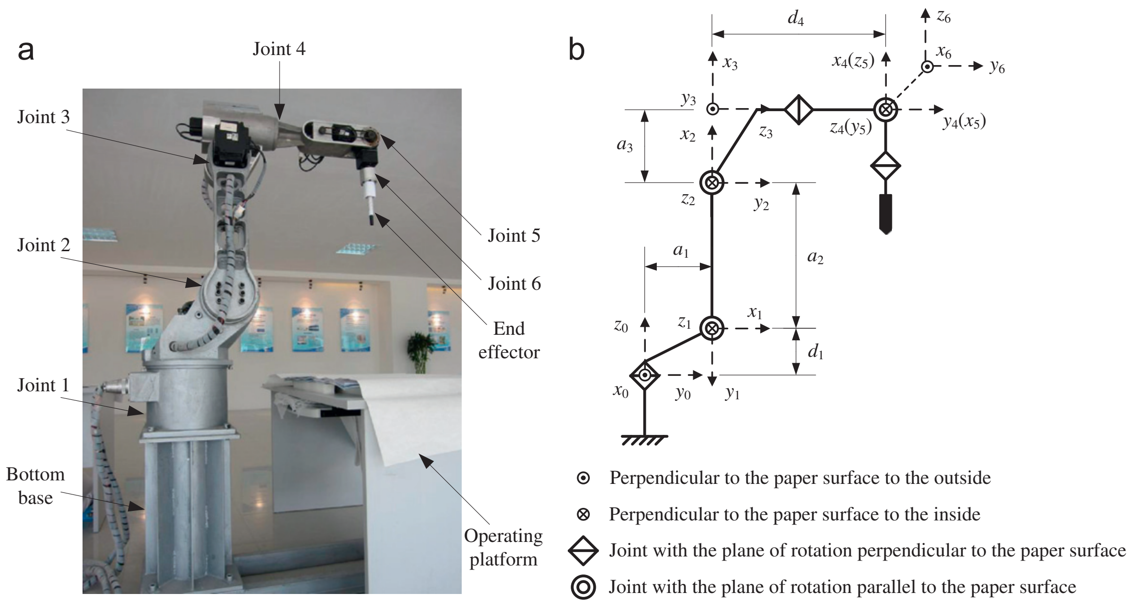

a
Joint 4
Joint 3
Joint 2
Joint 1
Bottom base
Joint 5
Joint 6
End effector
Operating platform
b
d_4
z_6
x_6
y_6
x_3
$x_4(z_5)$
y_3
z_3
$z_4(y_5)$
$y_4(x_5)$
x_2
a_3
z_2
y_2
a_1
a_2
z_0
z_1
x_1
d_1
x_0
y_0
y_1
Perpendicular to the paper surface to the outside
Perpendicular to the paper surface to the inside
Joint with the plane of rotation perpendicular to the paper surface
Joint with the plane of rotation parallel to the paper surface

Fig. 2. QJ-I (a) and its D-H link coordinate (b).

Table 1
D-H parameters and joint working range of QJ-I.

	d_i/mm	a_i/mm	α_i/(deg.)	θ_i	Initial value/(deg.)	Working range/(deg.)
1	250	150	−90	θ_1	90	[−40, 220]
2	0	550	0	θ_2	−90	[−190, 60]
3	0	160	−90	θ_3	0	[−126, 86]
4	594	0	90	θ_4	0	[−130, 130]
5	0	0	90	θ_5	90	[−102, 102]
6	0	0	0	θ_6	−90	(−270, 90]

coordinate shown in Fig. 2b, and the D-H parameters as well as the working range of each joint angle in Table 1. The task is to track a closed trajectory called QS Eagle (Fig. 3) in XY plane, which is made up of 136 discrete characteristic points. Taking the characteristic points of QS Eagle as sample points (where the ordinate value is set as $Z=303$ mm), we implement the trajectory planning in Cartesian space and joint space by the proposed method. Note that, to guarantee the closeness of QS Eagle profile, it is vital to add one more point which is the same as the start one to the end, i.e., 137 samples in total, to make the first and the last sample points coincident. The flow chart of the trajectory planning for QS Eagle is shown as in Fig. 4.

After successive calculations of the SQP equations, we get the optimal execution time of the tracking test as 68 s.

Particularly, when carrying out the trajectory planning in Cartesian space, we adopt the natural cubic spline as referred in Case 2 of Part 2, which ensures zero initial and ending acceleration for the end effector of QJ-I. The interpolated trajectories (include 681 discrete points along *X*/*Y* axis) and their derivatives up to the second-order are shown as in Figs. 5, 6 and 7, which indicate that the displacement, velocity, and acceleration along the coordinate axis *X*/*Y* are all continuous to time *t*.

And then, when implementing the trajectory planning in joint space, we apply the septuple B-splines in the interpolation, where

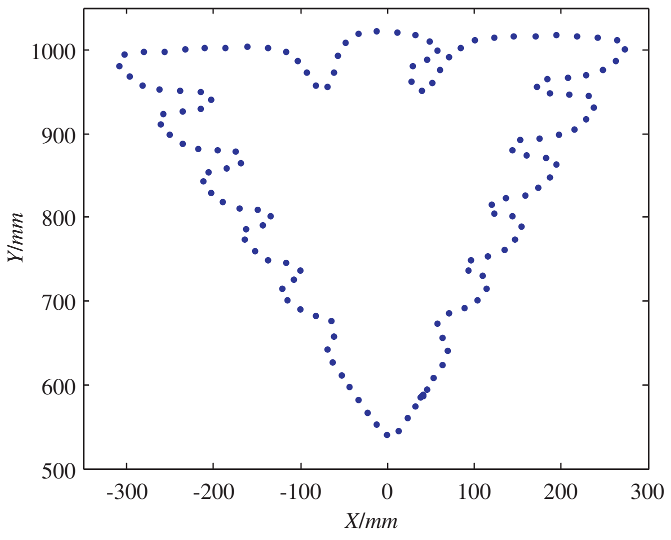
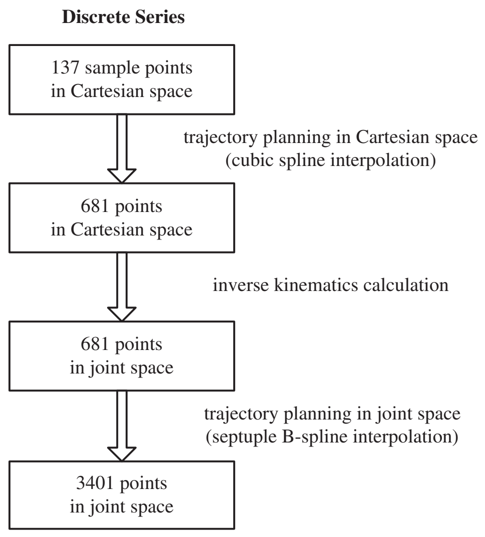


Fig. 3. Characteristic points of QS Eagle.



Discrete Series

137 sample points in Cartesian space

trajectory planning in Cartesian space (cubic spline interpolation)

681 points in Cartesian space

inverse kinematics calculation

681 points in joint space

trajectory planning in joint space (septuple B-spline interpolation)

3401 points in joint space

Fig. 4. Flow chart of trajectory planning for QS Eagle.

the value of angular velocity, acceleration and jerk at the initial and the ending moments for each joint are all configured as zero, and the kinematic constraints are given as in Table 2. The interpolated trajectories and their derivatives up to the third-order are shown as in Figs. 8, 9, 10 and 11, which indicate that the angle displacement, velocity, acceleration and jerk of each joint are all continuous to time t.

It is worth noting that, for a joint actuator, the needed output torque is in direct proportion to its angular acceleration. The continuous angular jerk of the joint actuator makes the change of the needed torque to time t continuous. This feature does much favor to reduce the flexible impact caused by exporting torques of joint actuators, and is vital for the stability and smoothness of the overall movements. Benefit from the septuple B-splines and the kinematic constraints of (24)$\sim$(26), the velocity, acceleration and jerk of the resulting trajectory for each robot joint are all both continuous and bounded. The improvement compared with the previous related literatures including Gasparetto and Zanotto [9] and Saravanan, Ramabalan, and Balamurugan [22] is that, continuous jerk is obtained by the proposed method, which is helpful to reduce the flexible impact caused by the joint actuator and enhance the fluency performance during the movement.

So far, we obtain a high smooth Joint Angular Position (q_i)—Time (t_i) discrete series of 3401 points with 20 ms time interval for each joint. In addition, the algorithm of trajectory planning including time optimization is executed on the PC (Fig. 1b). The discrete series of 3401 points are transformed into 3400 motion instructions, which are transferred to the Multi-Axis Motion Controller (MAMC) via a 32-bit PCI bus based interface, each instruction contains an angular displacement motion task that should be fulfilled in precise 20 ms (called instruction period). On the MAMC side, every displacement is interpolated into 50 micro-displacements, with execution time of 400us (called servo period) for each. Finally, each micro-displacement is implemented using a servo closed-loop motion control strategy one by one to complete the experimental test. The whole process of the movement of QJ-I is fluent and the result is shown as in Fig. 12.

Remark: When using cubic spline to interpolate a given trajectory in Cartesian space, the lineament of the trajectory curve will be changed to a certain extent. Therefore, in high precision situation, the density of sample points should be selected appropriately. If the sample points are too close, the cubic spline can not show its advantage. On the other side, if too

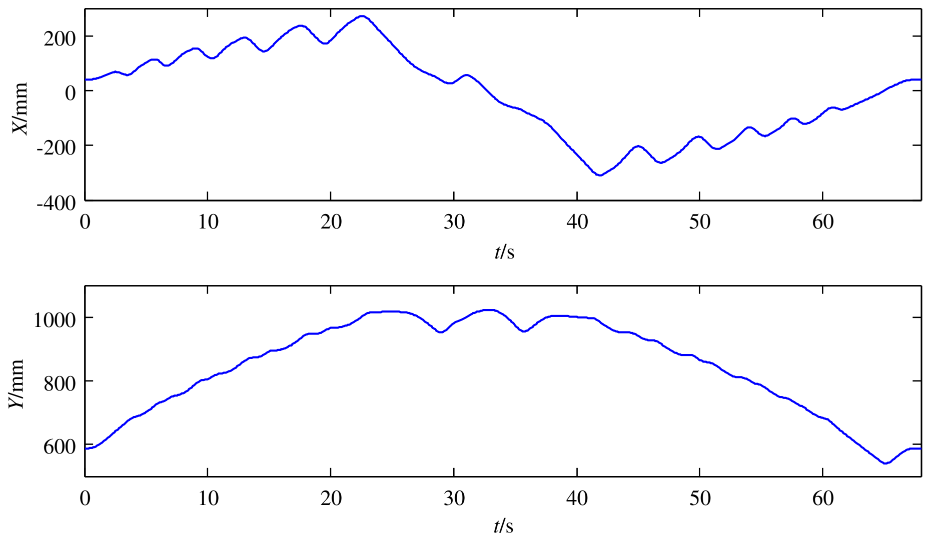


Fig. 5. Cubic spline interpolated curves of QS Eagle in Cartesian space.

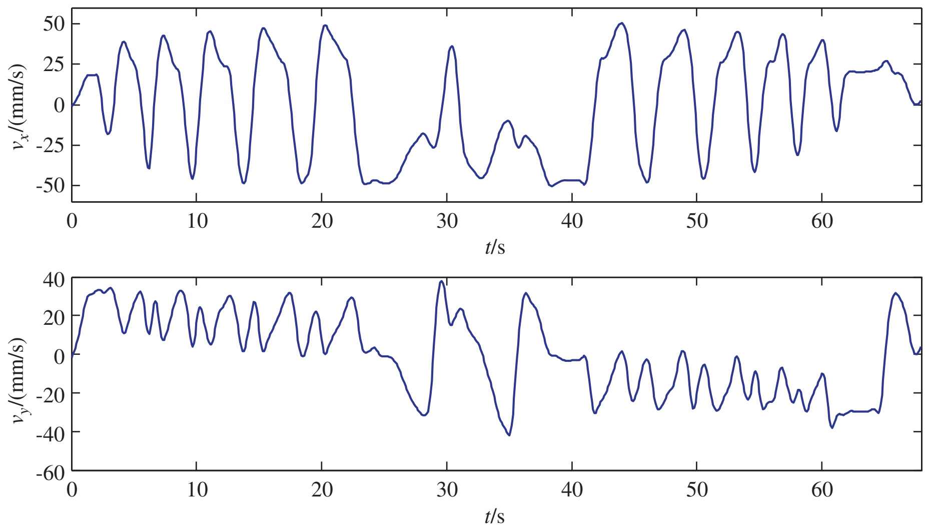


Fig. 6. First-order derivatives of cubic spline interpolated curves of QS Eagle in Cartesian space.

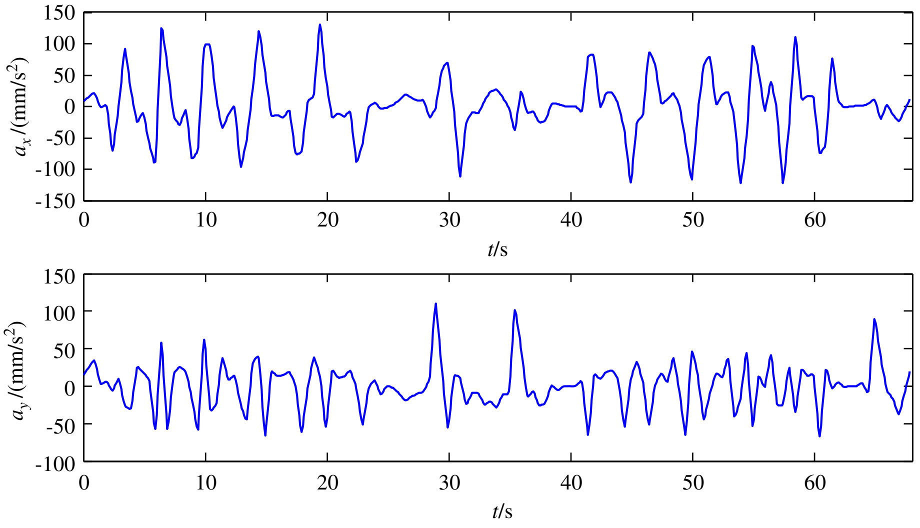


Fig. 7. Second-order derivatives of cubic spline interpolated curves of QS Eagle in Cartesian space.

Table 2
Kinematic constraints for each joint of QJ-I.

Joint no. m	Sup $\|V_m\|$ (deg./s)	Sup $\|A_m\|$ (deg./s^2)	Sup $\|J_m\|$ (deg./s^3)
1	100	45	60
2	95	40	60
3	100	75	55
4	150	70	70
5	130	90	75
6	110	80	70

sparse, the precision will be affected. Similar situation occurs in the septuple B-spline interpolating in the joint space. On most occasions, when setting the additional conditions to make the solution of the control points unique, we always tend to configure the value of the jerk at the initials and the endings as zero to enhance the stationarity of the movement. At this moment, if the interpolation points around are too dense, i.e., the time interval is too short, there will not have enough time for the jerk to complete the change due to the limited torque output performance of the joint actuators. Consequently, the tracking performance around the initials and the endings will be destroyed.

6. Conclusions

A time-optimal and jerk-continuous trajectory planning approach by combining the spline interpolating in Cartesian space and B-spline interpolating in joint space is proposed to gain a high smooth tracking performance in the practical motion task. Specially, when using cubic splines in the Cartesian space planning, the resulting trajectory is twice continuous differentiable, the initial and ending velocities or accelerations of the end effector of the robot manipulator can be set as needed. When invoking the septuple B-splines in the joint space planning, kinematic constraints and minimum-time optimization problem of the motion task are taken into account, the velocity, acceleration and jerk of the resulting trajectory are all bounded and continuous, which is advantageous to reducing the flexible impact and vibration caused by joint actuators. Moreover, the initial and ending values of the velocities, accelerations and jerks of all the robot joints can be configured almost arbitrarily. Experiments on a 6-DOF serial robot platform verify the effectiveness of the proposed method. Furthermore, it should be noted that, for any trajectory planning and optimization problems, the ultimate aim is to improve the performance and efficiency of the robot motion control. Taking the energy optimization into account in the proposed algorithm is worthy of further study and discussion.

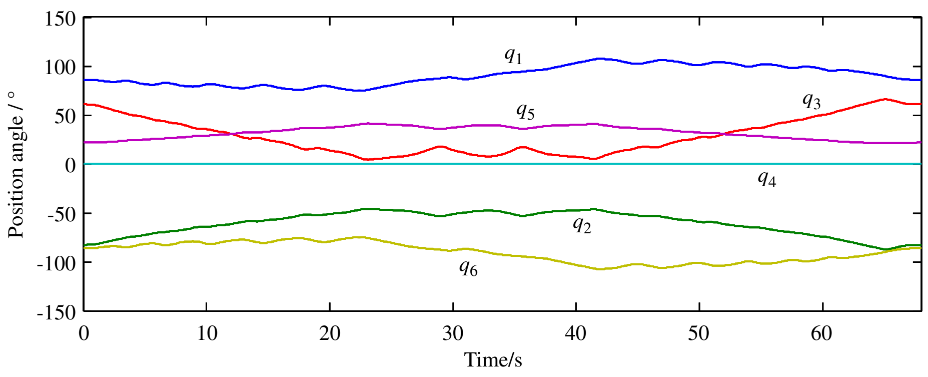


Fig. 8. Septuple B-spline interpolated curves of QS Eagle in joint space.

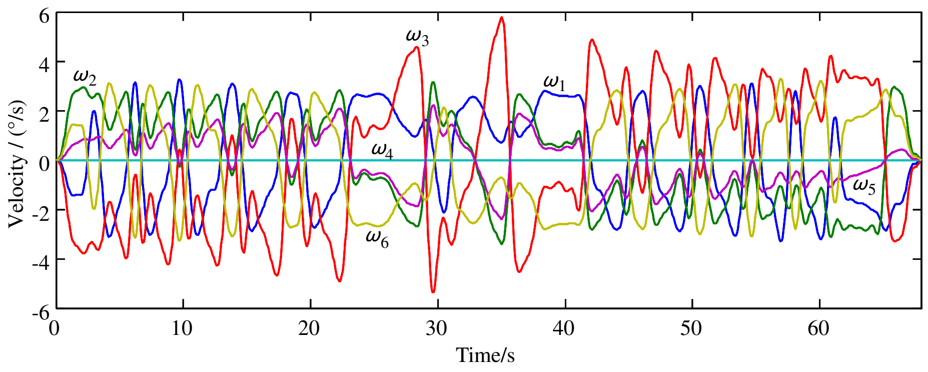


Fig. 9. First-order derivatives of septuple B-spline interpolated curves of QS Eagle in joint space.

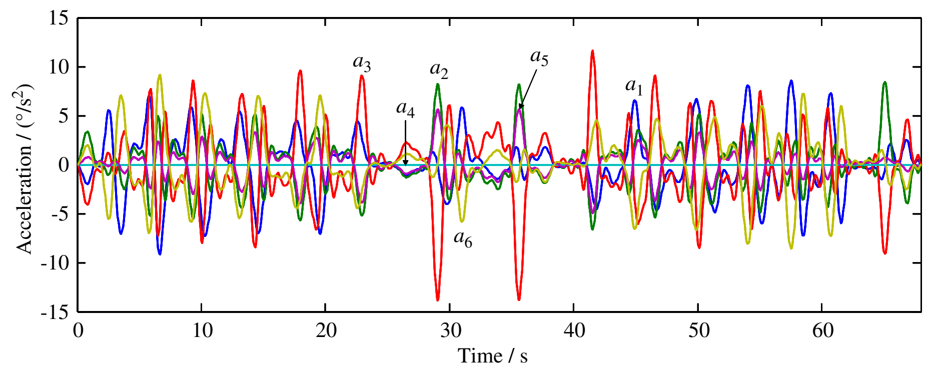


Fig. 10. Second-order derivatives of septuple B-spline interpolated curves of QS Eagle in joint space.

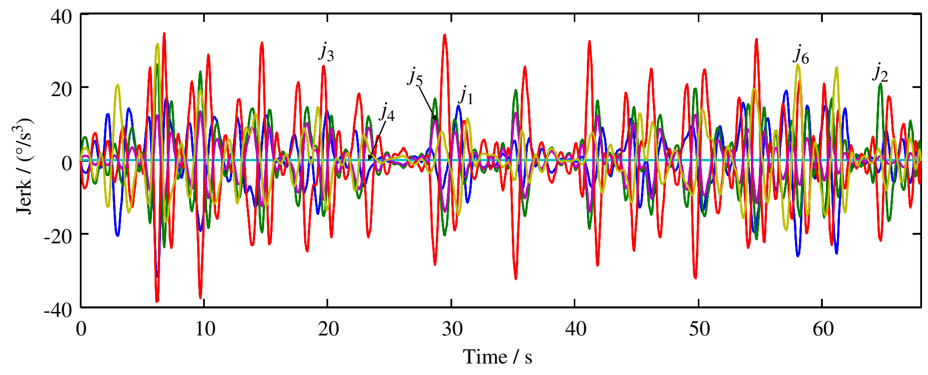


Fig. 11. Third-order derivatives of septuple B-spline interpolated curves of QS Eagle in joint space.

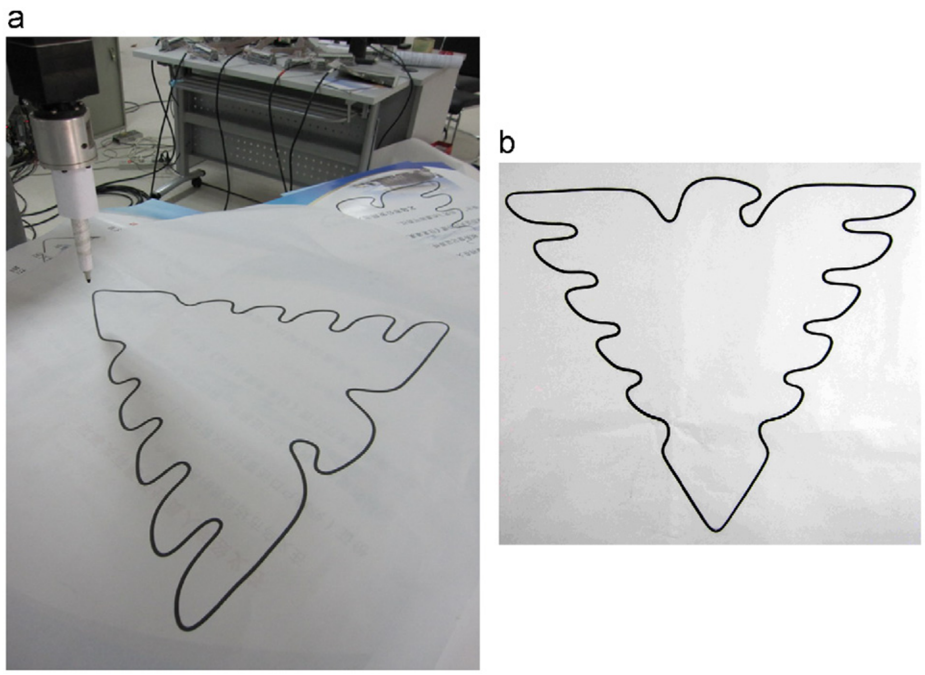


Fig. 12. Result of the QS Eagle tracking test.

Acknowledgments

We express our sincere thanks to the editors, referees and all the members of our discussion group for their beneficial comments. The work reported in this paper has been partially supported by the National Natural Science Foundation of China under Grant no. 61203337, the Zhejiang Province Natural Science Foundation of China under Grant no. LQ12F01004, the Fundamental Research Funds for the Central Universities under Grant nos. 11D10409, 12D10408 and the Young Teacher Training Program for the Shanghai Universities under Grant no. DHU11035.

Appendix A. Supplementary information

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.rcim.2012.08.002.

References

[1] Zheng CH, Su YX, Muller PC. Simple online smooth trajectory generations for industrial systems. Mechatronics 2009;19(4):571–6.
[2] Xu WF, Li C, Wang XQ, Liu Y, Liang B, Xu YS. Study on non-holonomic cartesian path planning of a free-floating space robotic system. Advanced Robotics 2009;23(1-2):113–43.
[3] Xian B, De Queiroz MS, Dawson D, Walker I. Task-space tracking control of robot manipulators via quaternion feedback. IEEE Transactions on Robotics and Automation 2004;20(1):160–7.
[4] Kant K, Zucker SW. Toward efficient trajectory planning—the path-velocity decomposition. International Journal of Robotics Research 1986;5(3):72–89.
[5] Stilman M. Global manipulation planning in robot joint space with task constraints. IEEE Transactions on Robotics 2010;26(3):576–84.
[6] Wu WX, Zhu SQ, Liu SG. Smooth joint trajectory planning for humanoid robots based on B-splines. In: Proceedings of the 2009 IEEE international conference on robotics and biomimetics; 2009:475–9.
[7] Chen MW, Zalzala AMS. Dynamic modelling and genetic-based trajectory generation for non-holonomic mobile manipulators. Control Engineering Practice 1997;5(1):39–48.
[8] Verscheure D, Demeulenaere B, Swevers J, De Schutter J, Diehl M. Time-optimal path tracking for robots: a convex optimization approach. IEEE Transactions on Automatic Control 2009;54(10):2318–27.
[9] Gasparetto A, Zanotto V. A new method for smooth trajectory planning of robot manipulators. Mechanism and Machine Theory 2007;42(4):445–71.
[10] Kröger T. On-line trajectory generation in robotic systems. Heidelberg: Springer; 2010.
[11] Niku SB. Introduction to robotics: analysis, systems, applications. Beijing: Publishing House of Electronics Industry; 2004.
[12] Biagiotti L, Melchiorri C. Trajectory planning for automatic machines and robots. Heidelberg: Springer; 2008.
[13] Zhou GB, Song BR, Xie JL. Numerical calculation. Beijing: Higher Education Press; 2008.
[14] Tsai MS, Nien HW, Yau HT. Development of a real-time look-ahead interpolation methodology with spline-fitting technique for high-speed machining. International Journal of Advanced Manufacturing Technology 2010;47(5-8):621–38.
[15] Ata AA. Optimal trajectory planning of manipulators: a review. Journal of Engineering Science and Technology 2007;2(1):32–54.
[16] Bazaz SF, Tondu B. Minimum time on-line joint trajectory generator based on low order spline method for industrial manipulators. Robotics and Autonomous Systems 1999;29(4):257–68.
[17] MacCarthy BL. Quintic splines for kinematic design. Computer-Aided Design 1988;20(7):406–15.
[18] Lin C, Chang P, Luh J. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. IEEE Transactions on Automatic Control 1983;28(12):1066–74.
[19] Gasparetto A, Zanotto V. Optimal trajectory planning for industrial robots. Advances in Engineering Software 2010;41(4):548–56.
[20] Gasparetto A, Zanotto V. A technique for time-jerk optimal planning of robot trajectories. Robotics and Computer-Integrated Manufacturing 2008;24(3):415–26.
[21] Thompson SE, Patel RV. Formulation of joint trajectories for industrial robots using B-splines. IEEE Transactions on Industrial Electronics 1987;34(2):192–9.
[22] Saravanan R, Ramabalan S, Balamurugan C. Evolutionary optimal trajectory planning for industrial robot with payload constraints. International Journal of Advanced Manufacturing Technology 2008;38(11-12):1213–26.
[23] Kröger T. On-line trajectory generation: straight-line trajectories. IEEE Transactions on Robotics 2011;27(5):1010–6.
[24] Kröger T, Wahl FM. On-line trajectory generation: basic concepts for instantaneous reactions to unforeseen events. IEEE Transactions on Robotics 2010;26(1):94–111.
[25] Kröger T. On-line trajectory generation: nonconstant motion constraints. In: Proceedings of the 2012 IEEE international conference on robotics and automation; 2012: p. 2048–54.
[26] Schumaker LL. Spline functions: basic theory. Cambridge: Cambridge University Press; 2007.
[27] De Boor CA. Practical guide to splines. New York: Springer; 2001.
[28] Jaluria Y. Computer methods for engineering with MATLAB applications. Bristol: Taylor & Francis; 2011.
[29] Nocedal J, Wright S. Numerical optimization. Berlin: Springer; 2006.