Journal of Computational and Applied Mathematics 329 (2018) 256–267

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Smooth orientation interpolation using parametric quintic-polynomial-based quaternion spline curve[☆]

Jieqing Tan [a], Yan Xing [a,*], Wen Fan [a], Peilin Hong [b]

[a] School of Mathematics, Hefei University of Technology, Hefei 230009, China
[b] Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230009, China

ARTICLE INFO

Article history:
Received 20 October 2016
Received in revised form 7 July 2017

Keywords:
Quaternion
G^2-continuity
Interpolation spline
Tension parameter

ABSTRACT

In this paper, a G^2continuous quintic-polynomial-based unit quaternion interpolation spline curve with tension parameters is presented to interpolate a given sequence of solid orientations. The curve in unit quaternion space S^3 is an extension of the quintic polynomial interpolation spline curve in Euclidean space. It preserves the interpolatory property and G^2 continuity. Meanwhile, the unit quaternion interpolation spline curve possesses the local shape adjustability due to the presence of tension parameters. The change of one tension parameter will only affect the adjacent two pieces of curves. Compared with the traditional B-spline unit quaternion interpolation curve and v-spline unit quaternion interpolation curve, the proposed curve can automatically interpolate the given data points, without solving the nonlinear system of equations over quaternions to obtain the control points, which greatly improves the computational efficiency. Simulation results demonstrate the effectiveness of the proposed scheme.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Quaternion curves have attracted considerable attention in the fields of computer animation, robot control and inertial navigation system. Unit quaternions are especially suitable for controlling the rotation of either 3D objects or virtual cameras [1,2]. One advantage of quaternions is that quaternions represent rotations more concisely than 3×3 rotation matrices because they only have 4 entries. So quaternions are more efficient than mainstream rotation matrices as computing the orientations. The product of two quaternions just means the composition of two successive rigid rotations. Compared with intuitive Euler angles, quaternions can avoid gimbal lock. Therefore, the design of unit quaternion curves has always become an interesting research topic [3–19].

In computer animation, it is a fundamental problem to generate the smooth motion of a rigid body interpolating a given sequence of key positions and orientations. A smooth rigid motion can be represented by two continuous curves: one is the position curve in the Euclidean space R^3, the other is the orientation curve in the rotation group $SO(3)$ [3,4]. They represent the translational and rotational motion of the rigid body respectively. We denote by S^3 the set of all unit quaternions, which forms a subgroup of the multiplication group of non-zero quaternions. Furthermore, the rotation group $SO(3)$ in R^3 can be obtained as a projective space of the unit 3-sphere S^3. The unit 3-sphere S^3 is the double cover of $SO(3)$. q and $-q$ map to the same element in $SO(3)$. Thus, the construction of orientation interpolation curve can be done in S^3, and the rotation control can be reduced to construct a unit quaternion curve in S^3.

☆ This work is supported by the National Natural Science Foundation of China under Grant No. 61472466 and No. 11601115.

* Corresponding author.
E-mail address: yanxing28@gmail.com (Y. Xing).

http://dx.doi.org/10.1016/j.cam.2017.07.007
0377-0427/© 2017 Elsevier B.V. All rights reserved.

Shoemake [2] constructed a splicing quaternion Bézier curve with C^1 continuity which interpolates a sequence of quaternions. Kim et al. [3,4] proposed a class of algebraic expressions to construct a variety of quaternion spline curves, and provided an iterative algorithm to solve the quaternionic non-linear system of equations to obtain the control points of the quaternion B-spline curve. Nielson [5] put forward unit quaternion ν-spline interpolatory curve, and also provided an iterative algorithm to acquire its control points. Ge et al. [6] developed an algorithm to search the control points by means of newly introduced several operational notations indicating quaternionic operations such as geodesic great circular arc interpolation $\oplus$ in S^3, and claimed the method can gain the accurate solution for quaternionic nonlinear system of equations. In the meanwhile, they presented the point insertion scheme which enables the interpolatory curve to be locally modifiable. Unfortunately, when solving the non-linear system of equations with the help of the newly introduced quaternion operators, they made the mistake that they took it for granted that the generalized addition operation $\oplus$ satisfies the associative law, which is not true usually. Su et al. [7] constructed an algebraic trigonometric blending quaternion interpolation spline curve, which can precisely interpolate a given sequence of solid orientations. However, the shape of the curve is not able to be adjusted locally.

The above analysis and discussion motivates us to look for a new type of quaternion interpolation spline curve which meets the needs of smoothness, local modifiability and computational efficiency. This paper mainly constructs a G^2-continuous parametric quintic polynomial based unit quaternion interpolation spline curve which automatically passes through a given sequence of data points without solving non-linear system of equations and preserves important geometric and differential properties of the analogous curve in R^3.

This paper is organized as follows. Section 1 describes the importance of quaternion curve and gives a brief review of previous methods. Section 2 introduces the operations and properties of quaternions. Section 3 first constructs the parametric quintic polynomial-based unit quaternion interpolation spline curve, then proves the interpolatory property and G^2 continuity of the curve. Section 4 shows and analyzes some experimental results. We conclude this paper in Section 5.

2. Operations and properties of quaternions

In this section, we simply recall the definition and operations of quaternions. Compared with a complex number $z = a + b\vec{i}\,(a, b \in R)$, a quaternion is defined by $q = w + x\vec{i} + y\vec{j} + z\vec{k}$ where $w, x, y, z \in R$ and $\vec{i}, \vec{j}, \vec{k}$ are three imaginary units, satisfying

$$\vec{i}^2 = \vec{j}^2 = \vec{k}^2 = -1, \qquad \vec{i}\vec{j} = -\vec{j}\vec{i} = \vec{k}, \qquad \vec{j}\vec{k} = -\vec{k}\vec{j} = \vec{i}, \qquad \vec{k}\vec{i} = -\vec{i}\vec{k} = \vec{j}. \tag{2.1}$$

The quaternion q can also be represented by a 4D vector (w, x, y, z) or an ordered pair $(s, \vec{v})$, where $s = w$ is a scalar, and $\vec{v} = (x, y, z)$ is a 3D vector.

Given two quaternions $q_n = w_n + x_n\vec{i} + y_n\vec{j} + z_n\vec{k}\,(n = 0, 1)$, the addition and subtraction of quaternions are defined by:

$$q_0 \pm q_1 = (w_0 \pm w_1) + (x_0 \pm x_1)\vec{i} + (y_0 \pm y_1)\vec{j} + (z_0 \pm z_1)\vec{k}. \tag{2.2}$$

The product of two quaternions is defined by

$$\begin{aligned} q_0q_1 = {} & (w_0w_1 - x_0x_1 - y_0y_1 - z_0z_1) + (w_0x_1 + x_0w_1 + y_0z_1 - z_0y_1)\vec{i} \\ & + (w_0y_1 + y_0w_1 + z_0x_1 - x_0z_1)\vec{j} + (w_0z_1 + z_0w_1 + x_0y_1 - y_0x_1)\vec{k}. \end{aligned} \tag{2.3}$$

Note that the multiplication is associative, but non-commutative.

The inner product of two quaternions is the same as the dot product of two 4D vectors.

$$(q_0, q_1) = w_0w_1 + x_0x_1 + y_0y_1 + z_0z_1. \tag{2.4}$$

The conjugate of a quaternion $q = w + x\vec{i} + y\vec{j} + z\vec{k}$ is defined by

$$\overline{q} = w - x\vec{i} - y\vec{j} - z\vec{k}, \tag{2.5}$$

$$\text{and} \quad \overline{q_1q_2} = \overline{q_2}\,\overline{q_1}. \tag{2.6}$$

The norm of a quaternion is

$$\|q\| = \sqrt{q\overline{q}} == \sqrt{w^2 + x^2 + y^2 + z^2}. \tag{2.7}$$

If $\|q\| = 1$, then q is a unit quaternion.

The set of all unit quaternions constitutes the space S^3.

The inverse of a quaternion is

$$q^{-1} = \frac{\overline{q}}{\|q\|^2}. \tag{2.8}$$

A unit or normalized quaternion $q = w + x\vec{i} + y\vec{j} + z\vec{k} \in S^3$, where $w^2 + x^2 + y^2 + z^2 = 1$, can be denoted as

$$q = \cos\theta + \vec{n}\sin\theta, \tag{2.9}$$

where $\theta = \arccos w \in [0, \pi]$ and $\vec{n} = \frac{(x,y,z)}{\sqrt{x^2+y^2+z^2}}$.

The exponential form for unit quaternion $q = \cos\theta + \vec{n}\sin\theta$ is

$$q = \exp(\theta\vec{n}), \tag{2.10}$$

where $\vec{n} \in S^2$ is a unit 3D vector, and $\theta \in [0, \pi]$.

The natural logarithm of a unit quaternion $q = \cos\theta + \vec{n}\sin\theta$ is

$$\log(q) = \vec{n} \cdot \theta, \tag{2.11}$$

where $\theta \in [0, \pi]$ and $\vec{n} \in S^2$.

The derivative of the function $q^{f(t)}$ with respect to t is

$$\frac{d}{dt}q^{f(t)} = f'(t)q^{f(t)}\log(q), \tag{2.12}$$

where base q is a unit quaternion and exponent $f(t)$ is a real-valued function $f : R \to R$.

Especially, when $f(t) = t$, we get

$$\frac{d}{dt}q^t = q^t\log(q). \tag{2.13}$$

The formula for spherical linear interpolation (Slerp) [2] from q_0 to q_1 along the geodesic path on the unit sphere S^3 is given by

$$Slerp(q_0, q_1; t) = q_0(q_0^{-1}q_1)^t, \tag{2.14}$$

or

$$Slerp(q_0, q_1; t) = \frac{\sin(1-t)\alpha}{\sin\alpha}q_0 + \frac{\sin t\alpha}{\sin\alpha}q_1 \tag{2.15}$$

where $\|q_0\| = \|q_1\| = 1$, $t \in [0, 1]$, and α is the angle between q_0 and q_1, satisfying $\cos\alpha = q_0 \cdot q_1$. The former Eq. (2.14) is obtained from the perspective of group structure, and the latter Eq. (2.15) from the point of view of 4-D geometry.

The rotation of a vector $\vec{v}$ around the axis $\vec{n}$ by the angle 2θ based on the right-hand rule can be expressed as

$$q\vec{v}q^{-1} = \exp(\theta\vec{n})\vec{v}\exp(-\theta\vec{n}) = (\cos\theta + \vec{n}\sin\theta)\vec{v}(\cos\theta - \vec{n}\sin\theta). \tag{2.16}$$

3. Unit quaternion interpolation spline curve based on parametric quintic polynomials

3.1. The construction of parametric quintic polynomial-based unit quaternion interpolation spline curve

Shoemake [2] presented the formula for Slerp which can produce an orientation curve interpolating two given poses represented by two unit quaternions. However, in the rigid motion control application, there are usually more than two key poses to connect. In order to obtain smooth interpolatory orientation curve, Shoemake [2], Kim [4], Nielson [5,8], Ge [6], Su [7], Schlag [10], Wang [12], Ramamoorthi [14], Buss [15], Yu [17], et al. gave different construction schemes for quaternion interpolation spline curves. Some of them cannot adjust the shape of the curve. Some of them need time-consuming iteration process to find the control points of curve which reduces the production efficiency. Here, we will propose a G^2-continuous quaternion interpolation spline curve with tension parameters which can not only automatically pass through the given keyframe orientations precisely without solving the non-linear system of equations to find the quaternionic control points, but also ensure the expected continuity, and the local shape controllability via modifying the tension parameters. This part will first introduce quintic polynomial interpolation spline curves with tension parameters in R^3, then present the construction of a G^2-continuous quaternion interpolation spline curve.

3.1.1. The construction of quintic polynomial interpolation spline curves with tension parameters in Euclidean space

Given data points $P_i \in \mathbb{R}^d (d = 2 \text{ or } 3, i = 1, \ldots, n)$ and tension parameters $\alpha_i \in \mathbb{R}\,(i = 1, \ldots, n)$, a curve passing through $\{P_i\}_{i=1}^n$ is constructed as follows.

Let $P_0 = P_1$, $P_{n+1} = P_n$. For $i = 1, \ldots, n-1$, the ith piece of quintic polynomial curve is defined as

$$p_i(t) \triangleq p_i(t; \alpha_i, \alpha_{i+1}) = \sum_{j=0}^{3} C_{i,j}(t; \alpha_i, \alpha_{i+1})P_{i+j-1}, \quad t \in [0, 1], \tag{3.1}$$

where

$$\begin{cases} C_{i,0}(t;\alpha_i,\alpha_{i+1}) = -\dfrac{\alpha_i}{5}B_1(t) - \dfrac{7\alpha_i}{20}B_2(t), \\ C_{i,1}(t;\alpha_i,\alpha_{i+1}) = B_0(t) + B_1(t) + \left(1-\dfrac{\alpha_i}{10}\right)B_2(t) + \dfrac{9\alpha_{i+1}}{20}B_3(t) + \dfrac{\alpha_{i+1}}{5}B_4(t), \\ C_{i,2}(t;\alpha_i,\alpha_{i+1}) = \dfrac{\alpha_i}{5}B_1(t) + \dfrac{9\alpha_i}{20}B_2(t) + \left(1-\dfrac{\alpha_{i+1}}{10}\right)B_3(t) + B_4(t) + B_5(t), \\ C_{i,3}(t;\alpha_i,\alpha_{i+1}) = -\dfrac{7\alpha_{i+1}}{20}B_3(t) - \dfrac{\alpha_{i+1}}{5}B_4(t), \end{cases} \tag{3.2}$$

where $B_i(t) = \binom{5}{i}t^i(1-t)^{5-i}$ $(i = 0, 1, \ldots, 5)$, $t \in [0, 1]$ are quintic Bernstein polynomials.

The quintic polynomial functions $C_{i,j}(t) \triangleq C_{i,j}(t;\alpha_i,\alpha_{i+1})$ $(j = 0, 1, 2, 3)$, $t \in [0, 1]$ are derived in the following way.

Consider four adjacent data points $P_{i-1}, P_i, P_{i+1}, P_{i+2}$ which will be interpolated. Suppose the curve segment $p_i(t;\alpha_i,\alpha_{i+1})$ between P_i and P_{i+1} is a polynomial curve. Then it can be expressed by a Bézier curve. In order to ensure the whole spline curve to achieve the second order continuity, we assume it is a quintic Bézier curve

$$p_i(t;\alpha_i,\alpha_{i+1}) = \sum_{j=0}^{5} B_j(t)V_j, \quad t \in [0, 1], \tag{3.3}$$

satisfying

$$\begin{cases} p_i(0;\alpha_i,\alpha_{i+1}) = V_0 = P_i, \\ p_i'(0;\alpha_i,\alpha_{i+1}) = 5(V_1 - V_0) = \alpha_i(P_{i+1} - P_{i-1}), \\ p_i''(0;\alpha_i,\alpha_{i+1}) = 20(V_2 - 2V_1 + V_0) = \alpha_i(P_{i+1} - 2P_i + P_{i-1}), \\ p_i(1;\alpha_i,\alpha_{i+1}) = V_5 = P_{i+1}, \\ p_i'(1;\alpha_i,\alpha_{i+1}) = 5(V_5 - V_4) = \alpha_{i+1}(P_{i+2} - P_i), \\ p_i''(1;\alpha_i,\alpha_{i+1}) = 20(V_5 - 2V_4 + V_3) = \alpha_{i+1}(P_{i+2} - 2P_{i+1} + P_i), \end{cases} \tag{3.4}$$

where $\{V_j\}_{j=0}^5$ are 6 control points for the quintic Bézier curve, α_i and α_{i+1} are tension parameters corresponding to data points P_i and P_{i+1}, and $\{B_j(t)\}_{j=0}^5$ are quintic Bernstein polynomials.

Solving for $\{V_j\}_{j=0}^5$ from the above system of equations (3.4), and substituting the results into (3.3), from $p_i(t;\alpha_i,\alpha_{i+1}) = \sum_{j=0}^5 B_j(t)V_j = \sum_{j=0}^3 C_{i,j}(t;\alpha_i,\alpha_{i+1})P_{i+j-1}$, we obtain (3.2).

It is not difficult to know

$$\begin{cases} p_i(1;\alpha_i,\alpha_{i+1}) = P_{i+1} = p_{i+1}(0;\alpha_{i+1},\alpha_{i+2}), \\ p_i'(1;\alpha_i,\alpha_{i+1}) = \alpha_{i+1}(P_{i+2} - P_i) = p_{i+1}'(0;\alpha_{i+1},\alpha_{i+2}), \\ p_i''(1;\alpha_i,\alpha_{i+1}) = \alpha_{i+1}(P_{i+2} - 2P_{i+1} + P_i) = p_{i+1}''(0;\alpha_{i+1},\alpha_{i+2}). \end{cases} \tag{3.5}$$

3.1.2. The construction of parametric quintic polynomial-based quaternion interpolation spline curves in S^3

According to Kim's general algebraic construction scheme for unit quaternion spline curves [3], we construct an analogue of the above introduced parametric quintic polynomial interpolation spline curves in unit quaternion space S^3.

Definition 1. Given n keyframe orientations $Q_i \in S^3 (i = 1, \ldots, n)$, monotonically increasing knot vector $U = (u_1, \ldots, u_n)$, and tension parameter vector $\alpha = (\alpha_1, \ldots, \alpha_n)$. Let $Q_0 = Q_1$, $Q_{n+1} = Q_n$. For $i = 1, \ldots, n-1$,the ith piece of quintic-polynomial-based quaternion interpolation spline curve is defined as

$$q_i(u) = Q_{i-1}^{\tilde{C}_{i,0}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)} \prod_{j=1}^{3} (Q_{i+j-2}^{-1}Q_{i+j-1})^{\tilde{C}_{i,j}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}, \quad u \in [u_i, u_{i+1}], \tag{3.6}$$

where

$$\tilde{C}_{i,j}(t;\alpha_i,\alpha_{i+1}) = \sum_{k=j}^{3} C_{i,k}(t;\alpha_i,\alpha_{i+1}) \quad (j = 0, 1, 2, 3), \tag{3.7}$$

and $C_{i,k}(t;\alpha_i,\alpha_{i+1})$ $(k = 0, 1, 2, 3)$ is defined in Eqs. (3.2).

It is easy to know

$$q_i(u_i) = Q_i \quad \text{and} \quad q_i(u_{i+1}) = Q_{i+1}.$$

So the whole quaternion spline curve $q(u)$ interpolates the given n keyframe orientations.

3.2. The continuity of the quaternion interpolation spline curve

The quintic polynomial-based unit quaternion interpolation spline curve with tension parameters preserves many important properties of parametric quintic polynomial interpolation spline curve in R^3, such as interpolatory property, G^2-continuity and local shape modifiability. In the following, we give the proof of interpolatory property and G^2-continuity of the quaternion curve.

Because the proposed unit quaternion curve is composed of $n-1$ quintic polynomial-based quaternion curve segments $q_i(t)\ (i=1,\ldots,n-1)$, and each segment is the product of the composite functions of smooth polynomial functions and smooth exponential functions, so it suffices to study the continuity of the splicing curve at the connecting places, known as the internal knots.

Theorem 1. *Given a sequence of n keyframe orientations $\{Q_i\}_{i=1}^{n}$. Let $Q_0=Q_1$, $Q_{n+1}=Q_n$. Then the curve defined in Definition 1 interpolates the given sequence of orientations, and is G^2 continuous.*

Proof. Since $\tilde{C}_{i,0}(t;\alpha_i,\alpha_{i+1})=\sum_{j=0}^{3}C_{i,j}(t;\alpha_i,\alpha_{i+1})=\sum_{k=0}^{5}B_k(t)=1$, for $i=1,\ldots,n-1$, the ith quintic-polynomial-based unit quaternion spline curve segment $q_i(u)\ (u\in[u_i,u_{i+1}])$ can be represented by

$$q_i(u)=Q_{i-1}(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}. \tag{3.8}$$

Its first derivative is

$$\begin{aligned}q_i'(u)={}&Q_{i-1}\left[(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\\&+Q_{i-1}(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\\&+Q_{i-1}(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]',\end{aligned} \tag{3.9}$$

where

$$\left[(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'=(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left(\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)'\log(Q_{i-1}^{-1}Q_i),$$

$$\left[(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'=(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left(\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)'\log(Q_i^{-1}Q_{i+1}),$$

$$\left[(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'=(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left(\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)'\log(Q_{i+1}^{-1}Q_{i+2}).$$

And its second derivative is

$$\begin{aligned}q_i''(u)={}&Q_{i-1}\left[(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]''(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\\&+Q_{i-1}(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]''(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\\&+Q_{i-1}(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]''\\&+2Q_{i-1}\left[(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'\left[(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\\&+2Q_{i-1}\left[(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'\\&+2Q_{i-1}(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'\left[(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]',\end{aligned} \tag{3.10}$$

where

$$\begin{aligned}\left[(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]''={}&(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[\left(\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)'\log(Q_{i-1}^{-1}Q_i)\right]^2\\&+(Q_{i-1}^{-1}Q_i)^{\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left(\tilde{C}_{i,1}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)''\log(Q_{i-1}^{-1}Q_i),\end{aligned}$$

$$\left[(Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'' = (Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[\left(\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)'\log(Q_i^{-1}Q_{i+1})\right]^2 + (Q_i^{-1}Q_{i+1})^{\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left(\tilde{C}_{i,2}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)''\log(Q_i^{-1}Q_{i+1}),$$

$$\left[(Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\right]'' = (Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left[\left(\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)'\log(Q_{i+1}^{-1}Q_{i+2})\right]^2 + (Q_{i+1}^{-1}Q_{i+2})^{\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)}\left(\tilde{C}_{i,3}\left(\frac{u-u_i}{u_{i+1}-u_i};\alpha_i,\alpha_{i+1}\right)\right)''\log(Q_{i+1}^{-1}Q_{i+2}).$$

By direct computation, we have

$$\tilde{C}_{i,1}(0;\alpha_i,\alpha_{i+1}) = 1, \qquad \tilde{C}_{i,2}(0;\alpha_i,\alpha_{i+1}) = 0, \qquad \tilde{C}_{i,3}(0;\alpha_i,\alpha_{i+1}) = 0,$$

$$\tilde{C}_{i,1}(1;\alpha_i,\alpha_{i+1}) = 1, \qquad \tilde{C}_{i,2}(1;\alpha_i,\alpha_{i+1}) = 1, \qquad \tilde{C}_{i,3}(1;\alpha_i,\alpha_{i+1}) = 0,$$

$$\tilde{C}'_{i,1}(0;\alpha_i,\alpha_{i+1}) = \frac{\alpha_i}{u_{i+1}-u_i}, \qquad \tilde{C}'_{i,2}(0;\alpha_i,\alpha_{i+1}) = \frac{\alpha_i}{u_{i+1}-u_i}, \qquad \tilde{C}'_{i,3}(0;\alpha_i,\alpha_{i+1}) = 0,$$

$$\tilde{C}'_{i,1}(1;\alpha_i,\alpha_{i+1}) = 0, \qquad \tilde{C}'_{i,2}(1;\alpha_i,\alpha_{i+1}) = \frac{\alpha_{i+1}}{u_{i+1}-u_i}, \qquad \tilde{C}'_{i,3}(1;\alpha_i,\alpha_{i+1}) = \frac{\alpha_{i+1}}{u_{i+1}-u_i},$$

$$\tilde{C}''_{i,1}(0;\alpha_i,\alpha_{i+1}) = \frac{-\alpha_i}{(u_{i+1}-u_i)^2}, \qquad \tilde{C}''_{i,2}(0;\alpha_i,\alpha_{i+1}) = \frac{\alpha_i}{(u_{i+1}-u_i)^2}, \qquad \tilde{C}''_{i,3}(0;\alpha_i,\alpha_{i+1}) = 0,$$

$$\tilde{C}''_{i,1}(1;\alpha_i,\alpha_{i+1}) = 0, \qquad \tilde{C}''_{i,2}(1;\alpha_i,\alpha_{i+1}) = \frac{-\alpha_{i+1}}{(u_{i+1}-u_i)^2}, \qquad \tilde{C}''_{i,3}(1;\alpha_i,\alpha_{i+1}) = \frac{\alpha_{i+1}}{(u_{i+1}-u_i)^2}.$$

Substituting u_i or u_{i+1} for u in Eqs. (3.8)–(3.10) respectively, we get the following three pairs of equations.

$$\begin{cases} q_i(u_i) = Q_i, \\ q_i(u_{i+1}) = Q_{i+1}; \end{cases} \tag{3.11}$$

$$\begin{cases} q_i'(u_i) = \dfrac{\alpha_i}{u_{i+1}-u_i}Q_i[\log(Q_{i-1}^{-1}Q_i) + \log(Q_i^{-1}Q_{i+1})], \\ q_i'(u_{i+1}) = \dfrac{\alpha_{i+1}}{u_{i+1}-u_i}Q_{i+1}[\log(Q_i^{-1}Q_{i+1}) + \log(Q_{i+1}^{-1}Q_{i+2})]; \end{cases} \tag{3.12}$$

$$\begin{cases} q_i''(u_i) = \dfrac{\alpha_i}{(u_{i+1}-u_i)^2}Q_i\{\alpha_i[\log(Q_{i-1}^{-1}Q_i) + \log(Q_i^{-1}Q_{i+1})]^2 + [\log(Q_i^{-1}Q_{i+1}) - \log(Q_{i-1}^{-1}Q_i)]\}, \\ q_i''(u_{i+1}) = \dfrac{\alpha_{i+1}}{(u_{i+1}-u_i)^2}Q_{i+1}\{\alpha_{i+1}[\log(Q_i^{-1}Q_{i+1}) + \log(Q_{i+1}^{-1}Q_{i+2})]^2 + [\log(Q_{i+1}^{-1}Q_{i+2}) - \log(Q_i^{-1}Q_{i+1})]\}. \end{cases} \tag{3.13}$$

It can be seen from Eq. (3.11) that the ith piece of curve $q_i(u) (i = 1, \ldots, n-1)$ passes through Q_i and Q_{i+1}. So, the whole spline curve interpolates the given data points.

Let $\Delta u_i = u_{i+1} - u_i (i = 1, \ldots, n-1)$. Straightforward computations give that for $i = 1, \ldots, n-1$,

$$\begin{cases} q_i(u_{i+1}) = Q_{i+1} = q_{i+1}(u_{i+1}), \\ q_i'(u_{i+1}) = \dfrac{\Delta u_{i+1}}{\Delta u_i}q_{i+1}'(u_{i+1}), \\ q_i''(u_{i+1}) = \left(\dfrac{\Delta u_{i+1}}{\Delta u_i}\right)^2 q_{i+1}''(u_{i+1}). \end{cases} \tag{3.14}$$

Hence, the quintic-polynomial-based quaternion spline curve does interpolate the given data points $\{Q_i\}_{i=1}^n$, and is of G^2 continuity. This completes the proof. ◀

With uniform knot spacing $u_{i+1} - u_i = \Delta u_i = h$ for all $i = 1, \ldots, n-1$, it is easy to know from Eqs. (3.14) that the quintic-polynomial-based quaternion interpolation spline curve can achieve C^2 continuity at the internal knots u_i $(i = 2, \ldots, n-1)$.

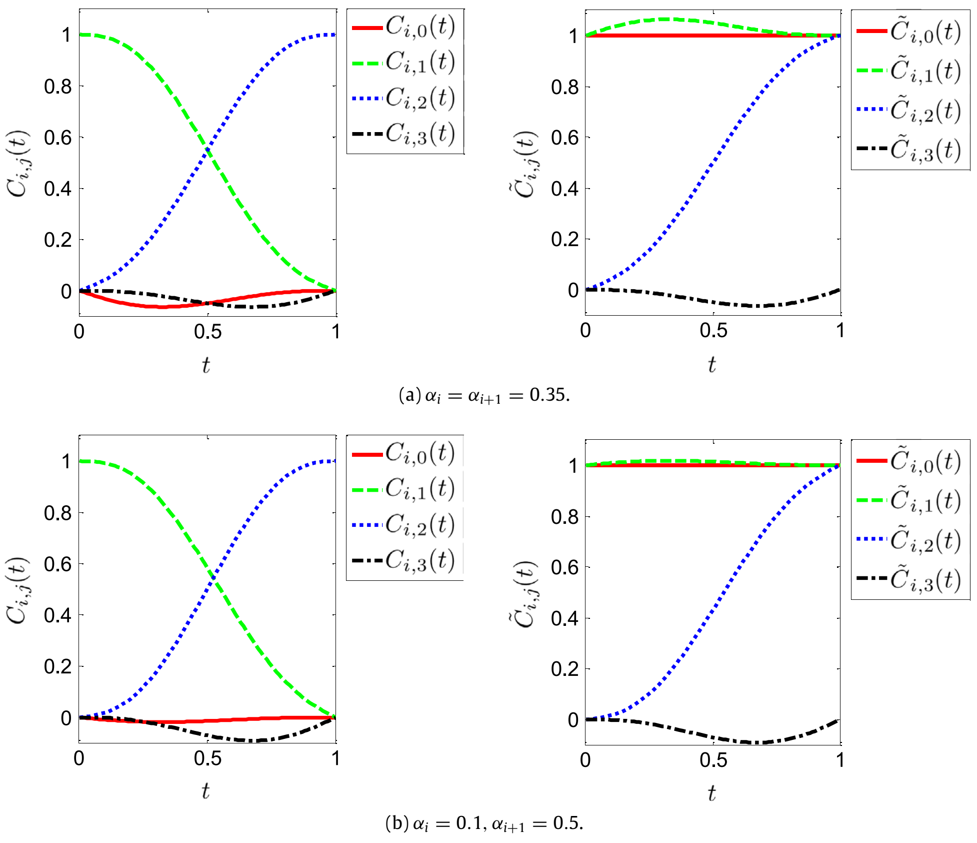


Fig. 1. The proposed quintic polynomial functions (left) and their cumulative forms (right).

4. Experiments

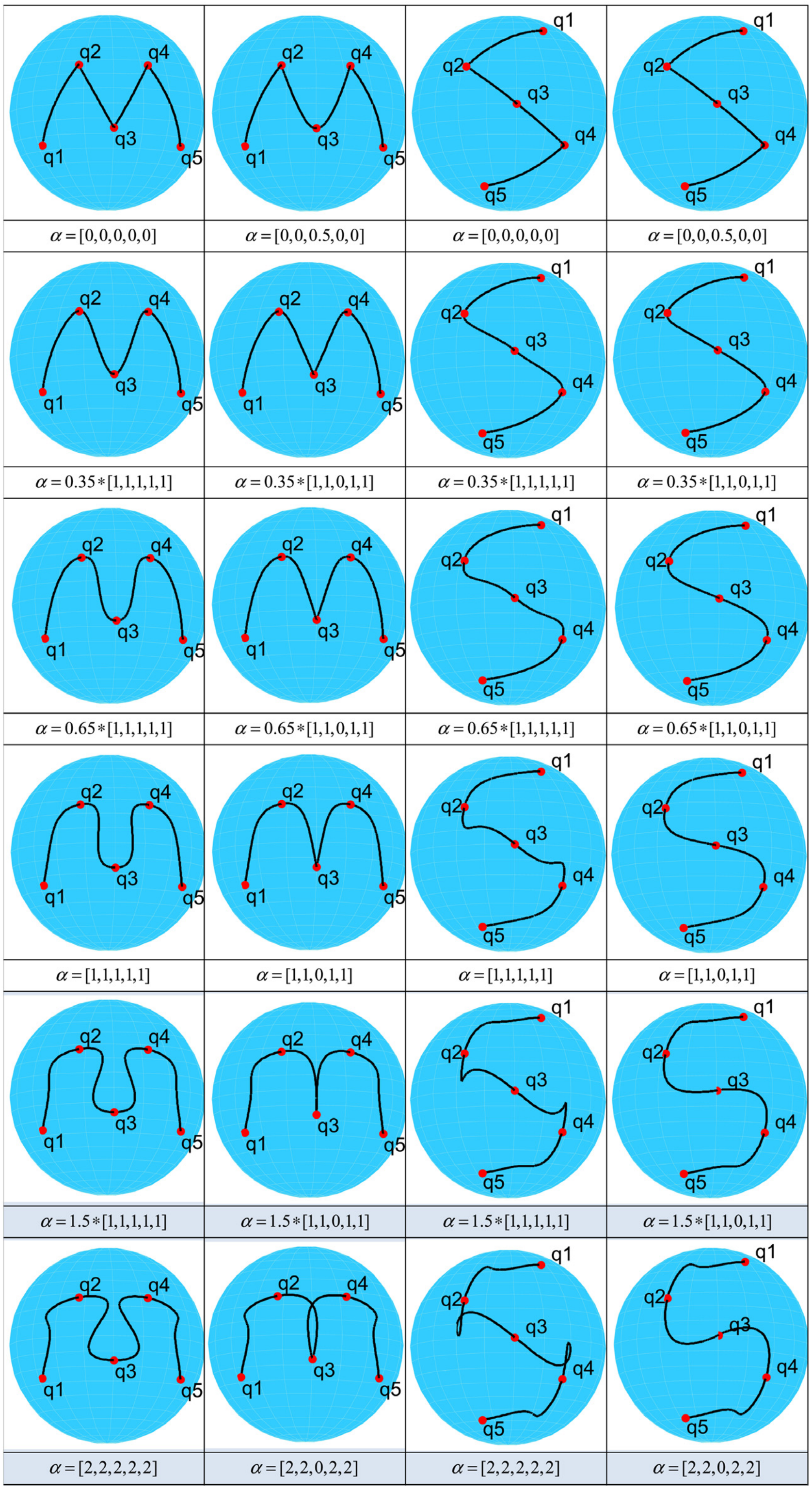
In this section, some experimental results are shown to demonstrate the effectiveness of our scheme.

First, the quintic polynomial functions $C_{i,j}(t) \triangleq C_{i,j}(t; \alpha_i, \alpha_{i+1})$ $(j = 0, 1, 2, 3)$ and the cumulative forms of functions $\tilde{C}_{i,j}(t) \triangleq \tilde{C}_{i,j}(t; \alpha_i, \alpha_{i+1}) = \sum_{k=j}^{3} C_{i,k}(t; \alpha_i, \alpha_{i+1})$ $(j = 0, 1, 2, 3)$ are depicted in Fig. 1, where $\alpha_i = \alpha_{i+1} = 0.35$ in (a) and $\alpha_i = 0.1, \alpha_{i+1} = 0.5$ in (b).

From Eq. (3.6), we know that each piece of curve $q_i(t)$ is only affected by two specific tension parameters α_i and α_{i+1}. In other words, one parameter only has an effect on the adjacent two pieces of curves. As shown in every row of Fig. 2, when only one tension parameter is modified, just the two pieces of curve incident to the corresponding knot change their shapes.

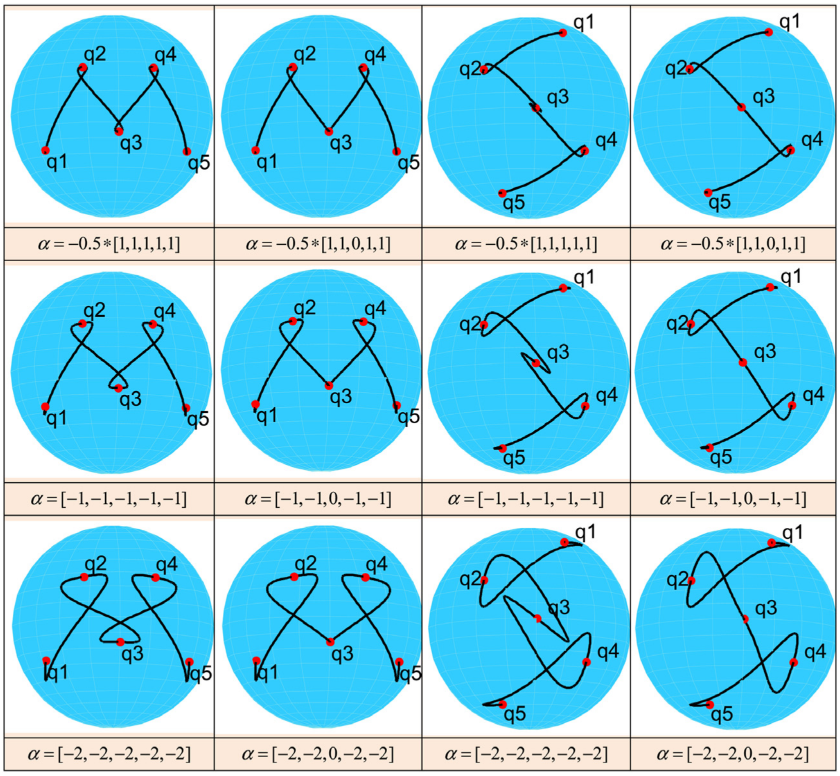
It is also observed from Fig. 2 that the larger the absolute value of a tension parameter is, the flatter the shape of the curve around the corresponding knot is. At this point, the larger the angular velocity and angular acceleration are (see Eqs. (3.12) and (3.13)), and the faster 3D object rotates around the corresponding knot (see Fig. 3(i)). When a tension parameter is close to 0, the curve is pretty sharp at the corresponding knot (see the case where $\alpha = [0, 0, 0, 0, 0]$ in Fig. 2), and because the angular velocity and angular acceleration are close to 0 (see Eqs. (3.12) and (3.13)), 3D object rotates very slow near the knot(see Fig. 3(j)). If a tension parameter is negative, there exists a kink or twist in the quaternion curve around the corresponding knot (see Fig. 2(b)). Generally, we choose non-negative tension parameters in [0,1].

As we know, when studying the problem of solid orientation interpolation, Kim et al. [4] presented an iterative algorithm to generate a unit quaternion B-spline interpolation curve. However, given the original keyframe orientations, the quaternion B-spline interpolation curve has fixed shape, and cannot adjust the rotation path according to the need. Nielson [5] proposed a geometric construction scheme of ν-spline quaternion curve which has tension parameters and variable knot spacing, but the acquisition of the control points for the curve needs time-consuming iteration operations. Su et al. [7] put forward an algebraic trigonometric blending quaternion interpolation spline curve which does not need iteration, but it cannot regulate the shape of curve due to the lack of shape parameters. Compared with these methods, our proposed scheme does not require an iterative process to solve the quaternionic nonlinear system of equations to obtain the control points of the spline curve, which saves time and improves the production efficiency, and at the same time it allows designers to adjust the shape of curve locally by changing the parameter value, and brings great flexibility in the design of smooth rigid motion.



(a) Tension parameter vector $\alpha \geq \mathbf{0}$.

Fig. 2. The impact of the tension parameters on the shape of curve with uniform knot spacing.



(b) Tension parameter vector $\alpha \leq \mathbf{0}$.

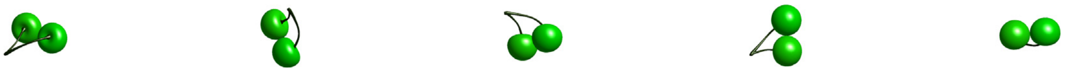
Fig. 2. (*continued*)

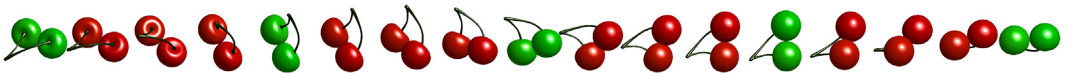
Fig. 3 shows 3D keyframe animations of cherries with the same keyframe orientations where position curves, for simplicity, are all straight lines, and orientation curves are unit quaternion B-spline interpolation curve [4], unit quaternion ν-spline curve [5], algebraic trigonometric blending interpolation (ATBI-) spline unit quaternion curve [7], and our proposed quintic-polynomial-based unit quaternion interpolation spline curves with different tension parameters, respectively. Here, Fig. 3(a) shows five key frames of cherries, Fig. 3(b) is the interpolation animation produced by quaternion B-spline curve, Fig. 3(c) is by quaternion ν-spline curve with uniform knot spacing and tension vector $\nu = \mathbf{0}$, Fig. 3(d) is by ATBI-spline quaternion curve, and Fig. 3(e)–(j) are built by our parametric C^2-continuous quintic-polynomial-based quaternion interpolation spline curves with uniform knot spacing where tension parameter vector α is [0.35, 0.35, 0.35, 0.35, 0.35], [0.6, 0.6, 0.6, 0.6, 0.6], [0.1, 0.6, 0, 0.6, 0.1], [1, 1, 1, 1, 1], [2, 2, 2, 2, 2] and [0, 0, 0, 0, 0], respectively. Five key frames in different poses are drawn in green, and the new produced frames are rendered in red.

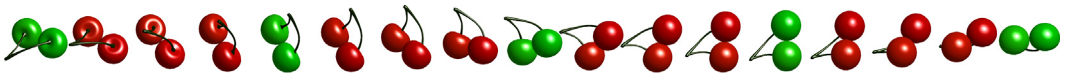
One important reason why we build the parametric quintic-polynomial-based quaternion interpolation spline curve is that the constructed curve possesses continuous curvature and local adjustability. The change of one tension parameter will affect the adjacent two segments. We usually choose the nonnegative tension parameter. The larger the tension parameter is, the faster the 3D object rotates near the corresponding knot, and vice versa. Furthermore, our quaternion curve can approximate the B-spline, ν-spline and ATBI-spline quaternion curves by adjusting the tension parameters.

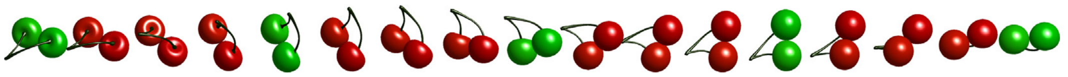
The high computation efficiency is the other important reason for us to construct the quaternion interpolation spline curve. Table 1 compares the elapsed time, the number of iterations, the parameter values and the construction type of B-spline, ν-spline, ATBI-spline unit quaternion curves and ours used in Fig. 3(b), (c), (d) and (f) respectively. Our scheme runs much faster than Nielson's scheme of ν-spline quaternion curve, and adds local shape adjustability compared with Kim's B-spline and Su's ATBI-spline unit quaternion curves. Note that the ATBI-spline quaternion curve [7] and our proposed unit quaternion interpolation spline curve are both defined with respect to the data points to be interpolated, rather than the control points, as in the case of B-spline curves [4], and the produced curves pass through the given sequence of orientation data automatically and accurately, therefore no iterations are needed and there is no approximation error.

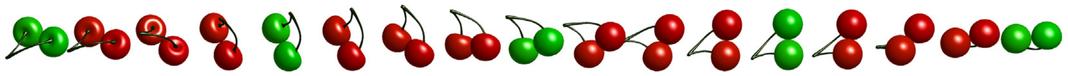
Fig. 4 compares the C^1-continuous cubic Hermite unit quaternion spline curve introduced in [3] and our C^2-continuous quintic unit quaternion interpolation spline curve with uniform parameter spacing. Although the teapot animations controlled by the two types of curves seem similar (see Fig. 4(c) and (d)), however, the curves are inherently different (see Fig. 4(a)). The proposed curve not only guarantees that each key frame orientation will be hit exactly and the tangents of the generated orientation curve are continuous over multiple segments, as Kim et al.'s cubic Hermite quaternion spline curve [3] can do, but also guarantees the second order derivatives and curvature over different segments are continuous.

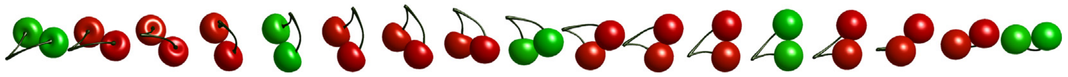

(a) Key frames of 3D animation.

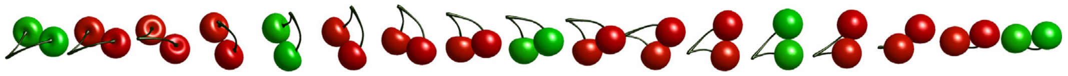

(b) Rotation controlled by Kim's B-spline quaternion curve.

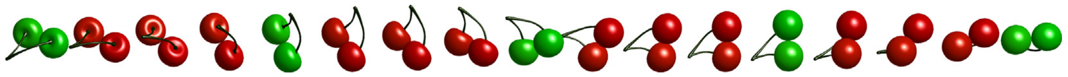

(c) Rotation controlled by Nielson's ν-spline quaternion curve.

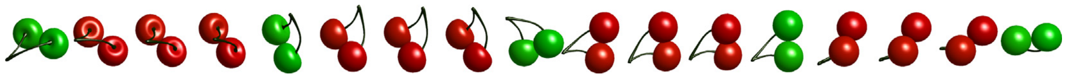

(d) Rotation controlled by Su's ATBI-spline quaternion curve.


(e) Rotation controlled by our proposed curve with $\alpha = [0.35, 0.35, 0.35, 0.35, 0.35]$.


(f) Rotation controlled by our proposed curve with $\alpha = [0.6, 0.6, 0.6, 0.6, 0.6]$.


(g) Rotation controlled by our proposed curve with $\alpha = [0.1, 0.6, 0, 0.6, 0.1]$.


(h) Rotation controlled by our proposed curve with $\alpha = [1, 1, 1, 1, 1]$.


(i) Rotation controlled by our proposed curve with $\alpha = [2, 2, 2, 2, 2]$.

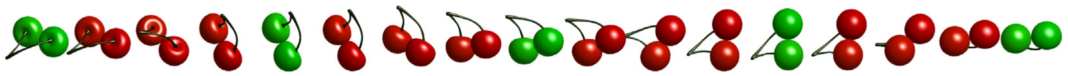

(j) Rotation controlled by our proposed curve with $\alpha = [0, 0, 0, 0, 0]$.

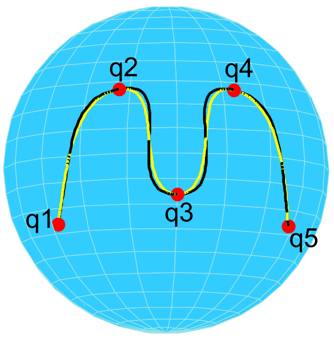
Fig. 3. Rigid motions controlled by different unit quaternion interpolation spline curves.

5. Conclusion

In this paper, we present a new scheme to construct unit quaternion interpolation spline curve which can generate a smooth orientation curve passing through a given sequence of key-frame poses. The parametric quintic-polynomial-based unit quaternion interpolation spline curve is constructed and proved to be G^2 continuity. Compared with B-spline and ν-spline unit quaternion interpolation curves, our scheme does not need solve a nonlinear system of equations and can precisely interpolate a given sequence of data points, which avoids the iteration process and improves the computation efficiency. Moreover, we introduced the tension parameters $\{\alpha_i\}_{i=1}^{n}$ which can locally modify the shape of the curve. The change of one parameter will only influence the shape of the two adjacent pieces of curves.

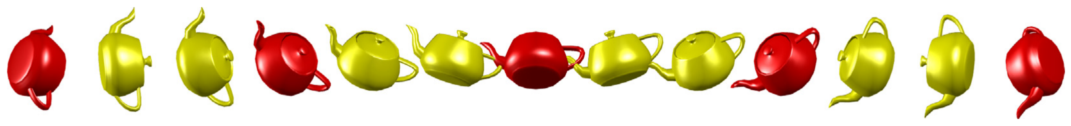
Table 1
Comparison of different unit quaternion interpolation spline curves.

Schemes	Number of the produced frames (given five key frames)	Elapsed time (s)	Number of iterations	Approximation error ($\leq$)	Tension parameter vector	Construction
Kim's cubic uniform B-spline quaternion curve [4]	401	0.360815	13	1e−6	No (fixed shape)	Algebraic construction by explicit formula
	17	0.355337	13			
Nielson's cubic ν-spline quaternion curve (uniform knots) [5]	401	6.744356	27	1e−6	$\nu = \mathbf{0}$	Geometric construction
	17	0.859442	27			
Su's algebraic trigonometric blending interpolation spline quaternion curve [7]	401	0.087366	–	–	No (fixed shape)	Algebraic construction by explicit formula
	17	0.086153				
Ours	401	0.087444	–	–	$\alpha = 0.6 * [1, 1, 1, 1, 1]$	Algebraic construction by explicit formula
	17	0.085934				



(a) Our C^2 quintic quaternion spline (in black) vs. the C^1 cubic Hermite quaternion spline (in yellow) $\alpha = [1, 1, 1, 1, 1]$.

(b) Key frames of teapot animation.


(c) Teapot motion controlled by cubic Hermite unit quaternion spline.

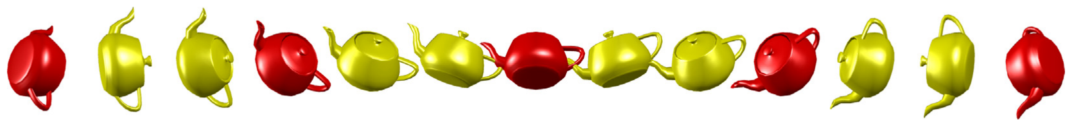

(d) Teapot motion controlled by our quintic unit quaternion interpolation spline curve.

Fig. 4. Comparison of our quintic quaternion spline and cubic Hermite quaternion spline.

There are some issues we should consider in the future, such as how to choose the tension parameters to achieve an optimal curve in the sense of curvature/torque energy minimization, and how to construct other forms of parameters to obtain a more intuitive, flexible and easy control of the orientations.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61472466 and 11601115. Special thanks also go to the referees.

References

[1] W.R. Hamilton, On quaternions, Proc. R. Ir. Acad. 3 (1847) 1–16.
[2] K. Shoemake, Animating rotation with quaternion curves, ACM SIGGRAPH Comput. Graph. 19 (3) (1985) 245–254.
[3] M.J. Kim, M.S. Kim, S.Y. Shin, A general construction scheme for unit quaternion curves with simple high order derivatives, in: Proceeding of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, ACM, 1995, pp. 369–376.
[4] M.J. Kim, M.S. Kim, S.Y. Shin, A C^2-continuous B-spline quaternion curve interpolating a given sequence of solid orientations, in: Proceeding of IEEE Computer Animation, 1995, pp. 72–81.
[5] G.M. Nielson, ν-quaternion splines for the smooth interpolation of orientations, IEEE Trans. Vis. Comput. Graphics 10 (2) (2004) 224–229.
[6] W. Ge, Z. Huang, G. Wang, Interpolating solid orientations with a C^2-continuous B-spline quaternion curve, in: 2nd International Conference on Technologies for E-Learning and Digital Entertainment, Hong Kong, Jun. 11-13, 2007. Book Series: Lecture Notes in Computer Science, Vol. 4469, pp. 606–615.
[7] B.Y. Su, J. Zhang, G.J. Wang, The solid orientations interpolation in quaternion space using a class of blending interpolation spline, Int. J. Adv. Comput. Technol. 5 (6) (2013) 335–341.
[8] G.M. Nielson, Smooth interpolation of orientations, in: Models and Techniques in Computer Animation, Springer, Japan, 1993, pp. 75–93.
[9] G.M. Nielson, R.W. Hieland, Animated rotations using quaternions and splines on a 4D sphere, Program. Comput. Softw. 18 (4) (1992) 145–154.
[10] J. Schlag, Using geometric constructions to interpolate orientation with quaternions, in: Graphics Gems II, 1991, pp. 377–380.
[11] J. Lee, S.Y. Shin, General construction of time-domain filters for orientation data, IEEE Trans. Vis. Comput. Graphics 8 (2) (2002) 119–128.
[12] W. Wang, B. Joe, Orientation interpolation in quaternion space using spherical biarcs, in: Proceedings of Graphics Interface. Canada, 1993, pp. 24–32.
[13] M.S. Kim, K.W. Nam, Interpolating solid orientations with circular blending quaternion curves, Comput. Aided Des. 27 (5) (1995) 385–398.
[14] R. Ramamoorthi, A.H. Barr, Fast construction of accurate quaternion splines, in: Proceeding of the 24th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison- Wesley Publishing Co., 1997, pp. 287–292.
[15] S.R. Buss, J.P. Fillmore, Spherical averages and applications to spherical splines and interpolation, ACM Trans. Graph. 20 (2) (2001) 95–126.
[16] E.L.G. Awad, E.R. Ebrahim, Exponential control of a rotational motion of a rigid body using quaternions, Appl. Math. Comput. 137 (2–3) (2003) 195–207.
[17] M.C. Yu, X.N. Yang, G.Z. Wang, Interpolation of unit quaternion curve with high order continuity, J. Comput. Aided Des. Comput. Graph. 17 (3) (2005) 437–441 (in Chinese).
[18] D. Han, G. Zeng, Y. Wang, A 2-DOF attitude control strategy based on quaternion, Adv. Sci. Lett. 11 (1) (2012) 443–447.
[19] Y. Xing, R.Z. Xu, J.Q. Tan, et al., A class of generalized B-spline quaternion curves, Appl. Math. Comput. 271 (2015) 288–300.