Journal of Computational and Applied Mathematics 229 (2009) 294–301

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Computing the minimum distance between two Bézier curves

Xiao-Diao Chen [a,c,d,*], Linqiang Chen [a], Yigang Wang [a], Gang Xu [a], Jun-Hai Yong [b], Jean-Claude Paul [b,c]

[a] Hangzhou Dianzi University, Hangzhou 310018, PR China
[b] School of Software, Tsinghua University, Beijing 100084, PR China
[c] INRIA, France
[d] State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, PR China

ARTICLE INFO

Article history:
Received 2 July 2008
Received in revised form 19 October 2008

Keywords:
Minimum distance
Bézier curve
Sweeping sphere clipping method

ABSTRACT

A sweeping sphere clipping method is presented for computing the minimum distance between two Bézier curves. The sweeping sphere is constructed by rolling a sphere with its center point along a curve. The initial radius of the sweeping sphere can be set as the minimum distance between an end point and the other curve. The nearest point on a curve must be contained in the sweeping sphere along the other curve, and all of the parts outside the sweeping sphere can be eliminated. A simple sufficient condition when the nearest point is one of the two end points of a curve is provided, which turns the curve/curve case into a point/curve case and leads to higher efficiency. Examples are shown to illustrate efficiency and robustness of the new method.

1. Introduction

The minimum distance computation problem between two object $\mathbf{O}_1$ and $\mathbf{O}_2$ can be described as to find the nearest point pair $(\mathbf{p}, \mathbf{q})$ such that $\mathbf{p} \in \mathbf{O}_1$, $\mathbf{q} \in \mathbf{O}_2$, and the distance between $\mathbf{p}$ and $\mathbf{q}$ is minimum. It is an important problem in many fields such as geometric modeling [4,8], computer graphics [5,11], and computer vision [4,8,11,13]. In geometric modeling, the distance information between a point and a curve or surface is essential for interactively selecting curves and surfaces. Distance information is needed for collision detection in CAD/CAM, computer graphics and computer vision. Moreover, the minimum distance information and the location where the minimum distance occurs are very useful in the location design of a bridge or undersea tunnel. The shorter the length of a bridge, the lower the cost and the less the corrasion of waves.

The minimum distance computation problem between curves and surfaces may be divided into five cases, i.e., point-curve, point–surface, curve–curve, curve–surface, surface–surface. All of these five cases can be turned into a root-finding problem of a non-linear equation system [5,9,10,15]. For example, for the curve/curve case, suppose that given two curves $\mathbf{C}_1(u)$ and $\mathbf{C}_2(v)$, when the nearest points are both inner points of the two curves, then we have the corresponding equation system

$$\begin{cases} S_u(u, v) = 0, \\ S_v(u, v) = 0, \end{cases} \tag{1}$$

where $S(u, v)$ is equal to $(\mathbf{C}_1(u) - \mathbf{C}_2(v))^2$, which is the squared distance between $\mathbf{C}_1(u)$ and $\mathbf{C}_2(v)$. Zhou et al. utilize both the projected-polyhedron and linear programming methods to solve the equation system. The methods on how to solve a

* Corresponding author at: Hangzhou Dianzi University, Hangzhou 310018, PR China.
E-mail address: xiaodiao@hdu.edu.cn (X.-D. Chen).

0377-0427/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2008.10.050

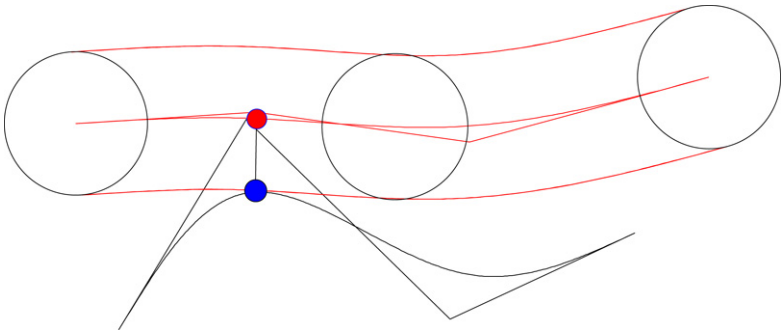

Fig. 1. Illustration of the sweeping sphere clipping method.

non-linear equation system can also be found in [3,10]. However, most of the roots are not mapping to the nearest points and are unnecessary to be solved, and the root-finding method itself may be sensitive to the input data. As Johnson points out in [5], the root-finding method has low robustness and low efficiency. In the worst case, all of the roots are unnecessary to compute (also see Fig. 3(a) in Section 3).

Limaiem has presented methods for the minimum distance computation problem between convex curves and surfaces, which converges to a local minimum distance by repeatedly finding the closest point on alternating curves or surfaces [6]. Lin utilizes bounding polyhedron to get the initial values for numerical methods to compute the minimum distance for the cases between concave surfaces [7]. Both approaches use the Newton–Raphson method to find the roots for some distance equations and need a good initial value for achieving the convergence result. However, this initial value is hard to obtain due to the complexity of the NURBS curve or surface's shape [8,11]. And even worse, the Newton–Raphson method may give a wrong answer [8].

The subdivision algorithms of NURBS curves and surfaces [12] are very robust. If one can find exclusion criteria whether a curve or surface can be pruned, the geometric pruning methods repeatedly testing the exclusion criteria and subdividing the curves or surfaces are able to meet with arbitrary tolerance and seem more geometric intuitive, reasonable and robust. Many geometric pruning methods cover the point-curve case and the point-surface case [1,2,8,11,14]. Piegl proposes to decompose the NURBS surface into quadrilaterals, and project the test point onto the nearest quadrilateral to get the initial value [11]. Ma provides the control-polygon-based method [8]. Selimovic improves these algorithms for the point-surface case by the tangent cone method [14]. However, these methods haven't covered the curve–curve case. The curve/curve case is important for the minimum distance computation problem between two surfaces. In fact, when the nearest points are on the boundary curves of the surfaces, it is indispensable to solve the minimum distance computation problem between two boundary curves.

This paper presents a sweeping sphere clipping method for computing the minimum distance computation between two Bézier curves. We first construct the sweeping sphere of a curve by rolling a sphere with its center point along the curve. If $(\mathbf{p}, \mathbf{q})$ is the nearest point pair, then all of the points on a curve except the nearest point $\mathbf{p}$ are outside the sweeping sphere of the other curve with its radius $\|\mathbf{pq}\|$; if all of the points on a curve except the nearest point $\mathbf{p}$ are outside the sweeping sphere of the other curve with its radius $\|\mathbf{pq}\|$, then $(\mathbf{p}, \mathbf{q})$ is the nearest point pair (see Fig. 1). Thus, the minimum distance computation between two Bézier curves is essentially equivalent to the intersection testing problem between a sweeping sphere and a curve. The initial radius of the sweeping sphere can be simply set as the distance between the end points of the two curves. From our experience, the efficiency of the new method may be improved by setting the initial radius of the sweeping sphere as the distance between an end-point on a curve and another curve. The nearest point on a curve must be contained in the sweeping sphere along the other curve, and all of the parts outside the sweeping sphere can be eliminated. A simple sufficient condition when the nearest point is one of the two end points of a curve is provided, which turns the curve/curve case into a point/curve case and leads to higher efficiency. Since a B-spline curve can be turned into several Bézier curves, the new method and its corresponding parallel algorithm also work with the B-spline curves.

The remaining paper is organized as follows. Section 2 explains the algorithm of the sweeping sphere clipping method. Section 3 shows several examples. Some conclusions are drawn at the end of the paper.

2. Algorithm of the sweeping sphere clipping method

Suppose that $B_n^i(u)$ and $B_m^j(v)$ are nth-degree and mth-degree Bézier basis functions, and that two Bézier curves are

$$\mathbf{C}_1(u) = \sum_{i=0}^{n} \mathbf{P}_i B_n^i(u), \quad \mathrm{u} \in [0, 1]$$

and

$$\mathbf{C}_2(v) = \sum_{j=0}^{m} \mathbf{Q}_j B_m^j(v), \quad \mathrm{v} \in [0, 1],$$

where $\{\mathbf{P}_i\}$ and $\{\mathbf{Q}_j\}$ are control points of the two curves in $\mathbb{R}^p$ space, $p \in \mathbb{Z}^+$. The key technique of the sweeping sphere clipping method is to judge whether a curve is outside of a sweeping sphere. It seems difficult to solve it directly by the information of the control points. We overcome this problem by analyzing the objective squared distance function

$$S(u, v) = \left(\sum_{i=0}^{n} \mathbf{P}_i B_n^i(u) - \sum_{j=0}^{m} \mathbf{Q}_j B_m^j(v) \right)^2 .$$

2.1. Computing $S(u, v)$ in Bézier form

Let C_i^j denotes the binomial coefficient $\binom{j}{i}$, $\theta = \max\{0, r-n\}$, $\upsilon = \min\{r, n\}$, $\sigma = \max\{0, k-m\}$, $\varsigma = \min\{k, m\}$, and

$$\begin{aligned} A_r &= \sum_{i=\theta}^{\upsilon} (\mathbf{P}_i \cdot \mathbf{P}_{r-i}) C_n^i C_n^{r-i} / C_{2n}^r, \\ B_k &= \sum_{j=\sigma}^{\varsigma} (\mathbf{Q}_j \cdot \mathbf{Q}_{k-j}) C_m^j C_m^{k-j} / C_{2m}^k, \end{aligned} \tag{2}$$

where $r = 0, 1, \ldots, 2n$, $k = 0, 1, \ldots, 2m$, and "$\cdot$" denotes the inner product between two vectors in $\mathbb{R}^p$ space. We obtain

$$\left(\sum_{i=0}^{n} \mathbf{P}_i B_n^i(u) \right)^2 = \sum_{r=0}^{2n} A_r B_{2n}^r(u) = \sum_{r=0}^{2n} A_r B_{2n}^r(u) \sum_{k=0}^{2m} B_{2m}^k(v),$$

$$\left(\sum_{j=0}^{m} \mathbf{Q}_j B_m^j(v) \right)^2 = \sum_{k=0}^{2m} B_k B_{2m}^k(v) = \sum_{k=0}^{2m} B_k B_{2m}^k(v) \sum_{r=0}^{2n} B_{2n}^r(u).$$

It can be verified that

$$\sum_{i=0}^{n} \mathbf{P}_i B_n^i(u) = \sum_{r=0}^{2n} \left(\sum_{i=\theta}^{\upsilon} \mathbf{P}_i C_n^i C_n^{r-i} / C_{2n}^r \right) B_{2n}^r(u)$$

and

$$\sum_{j=0}^{m} \mathbf{Q}_j B_m^j(v) = \sum_{k=0}^{2m} \left(\sum_{j=\sigma}^{\varsigma} \mathbf{Q}_j C_m^j C_m^{k-j} / C_{2m}^k \right) B_{2m}^k(v).$$

We have

$$\sum_{i=0}^{n} \mathbf{P}_i B_n^i(u) \cdot \sum_{j=0}^{m} \mathbf{Q}_j B_m^j(v) = \sum_{r=0}^{2n} \sum_{k=0}^{2m} C_{r,k} B_{2n}^r(u) B_{2m}^k(v),$$

where

$$C_{r,k} = \left(\sum_{i=\theta}^{\upsilon} \mathbf{P}_i C_n^i C_n^{r-i} / C_{2n}^r \right) \cdot \left(\sum_{j=\sigma}^{\varsigma} \mathbf{Q}_j C_m^j C_m^{k-j} / C_{2m}^k \right). \tag{3}$$

Thus, we can turn $S(u, v)$ into a Bézier form and obtain

$$S(u, v) = \sum_{r=0}^{2n} \sum_{k=0}^{2m} D_{r,k} B_{2n}^r(u) B_{2m}^k(v), \tag{4}$$

where $D_{r,k} = A_r + B_k - 2C_{r,k}$.

2.2. Algorithm with new exclusion criteria

From the convex hull property, we obtain

Property 1. *If $D_{r,k} \geq \alpha$ for all $0 \leq r \leq 2n$ and $0 \leq k \leq 2m$, then $S(u, v) \geq \alpha$, where $\alpha \in \mathbb{R}^+$. This means that curve $\mathbf{C}_1(u)$ is outside of the sweeping sphere of curve $\mathbf{C}_2(v)$ with its radius $\sqrt{\alpha}$.*

To detect whether one of the point in the nearest point pair is an end-point of a curve, we have

Property 2. *If there exists an integer $k = 0$ or $k = 2n$, such that for all $i \in \{0, 1, \ldots, 2n\}$ and $j \in \{0, 1, \ldots, 2m\}$, $D_{i,j} \geq D_{k,j}$, then $S(u, v)$ must reach the minimum value at the boundary $u = 0$ or $u = 1$, which means the end-point $\mathbf{P}_0$ or $\mathbf{P}_n$ is the nearest point on $\mathbf{C}_1(u)$. Similarly, if there exists an integer $k = 0$ or $k = 2m$, such that for all $i \in \{0, 1, \ldots, 2n\}$ and $j \in \{0, 1, \ldots, 2m\}$, $D_{i,j} \geq D_{i,k}$, then $S(u, v)$ must reach the minimum value at the boundary $v = 0$ or $v = 1$, which means the end-point $\mathbf{Q}_0$ or $\mathbf{Q}_m$ is the nearest point on $\mathbf{C}_2(v)$.*

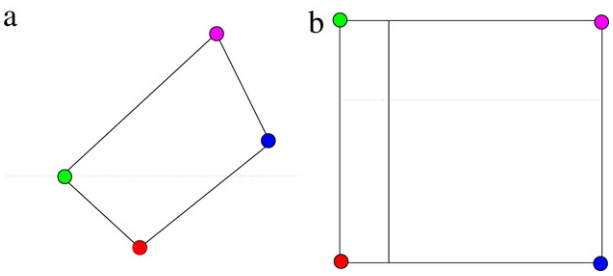

Fig. 2. Illustration of the heuristic method (a) the two adjacent corners of $S(u, v)$ with the smallest value are solid points in red and green, respectively; (b) $S(u, v)$ is subdivided at the solid line $u = i/(2n)$ in parametric domain. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Proof. Without loss of generality, suppose that $D_{i,j} \geq D_{0,j}$, for all $i \in \{0, 1, \ldots, 2n\}$ and $j \in \{0, 1, \ldots, 2m\}$. From the convex hull property, we obtain

$$\begin{aligned}
S(u, v) &= \sum_{r=0}^{2n}\sum_{k=0}^{2m} D_{r,k}B_{2n}^{r}(u)B_{2m}^{k}(v)\\
&\geq \sum_{r=0}^{2n}\sum_{k=0}^{2m} D_{0,k}B_{2n}^{r}(u)B_{2m}^{k}(v)\\
&= \sum_{r=0}^{2n}\sum_{k=0}^{2m} D_{0,k}B_{2m}^{k}(v)B_{2n}^{r}(u)\\
&= \sum_{r=0}^{2n}\left(\sum_{k=0}^{2m} D_{0,k}B_{2m}^{k}(v)\right)B_{2n}^{r}(u)\\
&= \left(\sum_{k=0}^{2m} D_{0,k}B_{2m}^{k}(v)\right)\sum_{r=0}^{2n} B_{2n}^{r}(u)\\
&= \sum_{k=0}^{2m} D_{0,k}B_{2m}^{k}(v).
\end{aligned}$$

So $S(u, v)$ must reach the minimum value at the boundary $u = 0$. □

From our experience, when the parametric domain is small (in our practical use within 10^{-3}), the new method may eliminate at least one of the two sub-surfaces after each subdivision. Usually it needs to solve one of the roots of the equation system (1) or none after the subdivision process. So about ten subdivision steps in U-direction may turns the U-direction interval of size δ_1 into a new interval of size $\delta_1/1000$. On the other hand, each iterative step with an initial value (u_0, v_0) in the Newton method needs to calculate the values of $S_u(u_0, v_0)$, $S_v(u_0, v_0)$, $S_{uu}(u_0, v_0)$, $S_{vv}(u_0, v_0)$, $S_{uv}(u_0, v_0)$, whose computation cost is about five times of that computation time of each subdivision of $S(u, v)$. In a word, the computation cost of four Newton iterative steps in the new method may turn the parameter domain of size $\delta_1 \times \delta_2$ into a new one of smaller size $(\delta_1/1000) \times (\delta_2/1000)$. In this paper, we don't use the Newton method, and just use the sweeping sphere clipping method combining with subdivision technique to solve the nearest points on the two curves, which leads to better robustness.

The algorithm of the sweeping sphere clipping method is as follows. It contains two mainly steps. The first step is to compute the Bézier form of $S(u, v)$ in Eq. (4). The second step is to compute the minimum value d of $S(u, v)$ as well as its parameter pair (u, v). Let $S_1(u, v)$ be the surface $S(u, v)$ or its surface patch. In the second step, we directly compute or omit the computation of the minimum value of $S_1(u, v)$ in the following three cases: (1) the parameter domain of the surface patch mapping to that of $S(u, v)$ is small enough in U or V-direction according to the tolerance ε, which means surface $S_1(u, v)$ is degenerated into a curve, then we compute the minimum value at the curve, which is also taken as the minimum value of $S_1(u, v)$; (2) Property 1 with the current minimum value α is satisfied for $S_1(u, v)$, which means that the minimum value of $S_1(u, v)$ is larger than α and is unnecessary to compute; (3) Property 2 is satisfied, which means the minimum value occurs at a boundary curve, then we compute the minimum value at the boundary curve, which is also the minimum value of $S_1(u, v)$.

In other cases, the minimum value of $S_1(u, v)$ may occurs at its inner point. Then we subdivide $S_1(u, v)$ into two surface patches $S_2(u, v)$ and $S_3(u, v)$. The following heuristic method for the subdivision process may lead to a better performance (see Fig. 2). Let $\{D_{r,k}^1\}$ be the control net of $S_1(u, v)$. Suppose $D_{i,j}^1$ is the minimum value of all $\{D_{r,k}^1\}$. Then $S_1(u, v)$ may be subdivided at $u = i/(2n)$ or $v = j/(2m)$. The direction is chosen such that the two adjacent corners of $S(u, v)$ with the

smallest value remain in the same part (also see [14]). It is obvious that

$$\min\{S_1(u, v)\} = \min\{\min\{S_2(u, v)\}, \min\{S_3(u, v)\}\}.$$

So we compute the minimum values of $S_2(u, v)$ and $S_3(u, v)$, and pick the smaller value between the two minimum values as the minimum value of $S_1(u, v)$. After some finite subdivision steps, at least one of the three above cases may be satisfied, then the minimum values of the surface patches can be directly computed. Thus, we finally obtain the minimum value of $S(u, v)$. The algorithms are as follows.

Algorithm 1. Algorithm for computing the minimum value of a surface patch $S_1(u, v)$.

Input: $S_1(u, v)$, the current minimum value α, the corresponding parameter domain **H** mapping to that of $S(u, v)$ and the tolerance ε.
Output: The minimum value d (only valid for the case $d < \alpha$) and its corresponding parameter pair (u, v) where d occurs as well;

(1) If **H** is small enough in U or V-direction according to the tolerance ε, goto Step 2;
Otherwise, goto Step 4;
(2) $S_1(u, v)$ may be taken as a curve $S_1(u_m, v)$ or $S_1(u, v_m)$, where u_m and v_m are the mid value of U and V-direction interval, respectively;
Compute the minimum value d of $S_1(u_m, v)$ or $S_1(u, v_m)$, record the parameter pair (u, v); goto Step 3;
(3) If d is smaller than α, then update $\alpha = d$ and its parameter pair as well;
Goto Step 9;
(4) Update $\alpha = \min\{\alpha, S_1(0, 0), S_1(1, 0), S_1(0, 1), S_1(1, 1)\}$;
goto Step 5;
(5) If Property 1 is satisfied, the minimum value of $S_1(u, v)$ is larger than α; Let $d = \alpha$, and goto Step 9;
Otherwise, goto Step 6;
(6) If Property 2 is satisfied, the minimum value of $S_1(u, v)$ occurs at a boundary curve;
Compute the minimum value d of $S_1(u, v)$ at the boundary curve, goto Step 7;
Otherwise, goto Step 8;
(7) If d is smaller than α, then update $\alpha = d$ and its parameter pair as well;
Goto Step 9;
(8) Subdivide $S_1(u, v)$ into two surface patches by using the heuristic method to obtain $S_2(u, v)$ and $S_3(u, v)$;
Record the corresponding parameter domains of $S_2(u, v)$ and $S_3(u, v)$;
Use the Algorithm 1 to compute the minimum value d_2 of $S_2(u, v)$ and its parameter pair as well;
Use the Algorithm 1 to compute the minimum value d_3 of $S_3(u, v)$ and its parameter pair as well;
Pick the smaller value d between d_2 and d_3 and the corresponding parameter pair as well;
Goto Step 9;
(9) **End of** Algorithm 1: Return the minimum value d and its parameter pair as well.

Algorithm 2. Algorithm for computing the minimum distance between two Bézier curves.

Input: Two Bézier curves.
Output: The nearest point pair as well as the minimum distance d.

1. Compute $S(u, v)$ in a Bézier form;
2. Set α as $\min\{S(0, 0), S(1, 0), S(0, 1), S(1, 1)\}$;
3. **End of** Algorithm 2: Use Algorithm 1 to compute the minimum value d of $S(u, v)$ and the parameter pair (u, v) where the minimum value occurs;
Compute the nearest point pair mapping to the parameter pair (u, v), and return the nearest points and its minimum distance d as well (see Fig. 5).

3. Analyses and examples

This section compares the new method with the method in [10]. Firstly, it needs to transform both $S_u(u, v)$ and $S_v(u, v)$ in the equation system (1) in [10] into the Bézier form while it only needs to transform $S(u, v)$ of Eq. (4) into the Bézier form in the new method, and the corresponding transformation time in [10] is nearly twice as that time in the new method.

Secondly, in each subdivision process, the method in [10] needs to subdivide both $S_u(u, v)$ and $S_v(u, v)$ while the new method only needs to subdivide $S(u, v)$. Suppose that n is equal to m, then the corresponding computation time on each subdivision process in the method in [10] is about two times of that time of the new method.

Finally, the new method can eliminate most of the roots of the equation system (1) in [10] by using Properties 1 and 2 during the subdivision process, and then the number of the remaining patches in the new method is much less than the total number of the solutions of the equation system in [10]. Thus, the subdivision time in the new method is much less

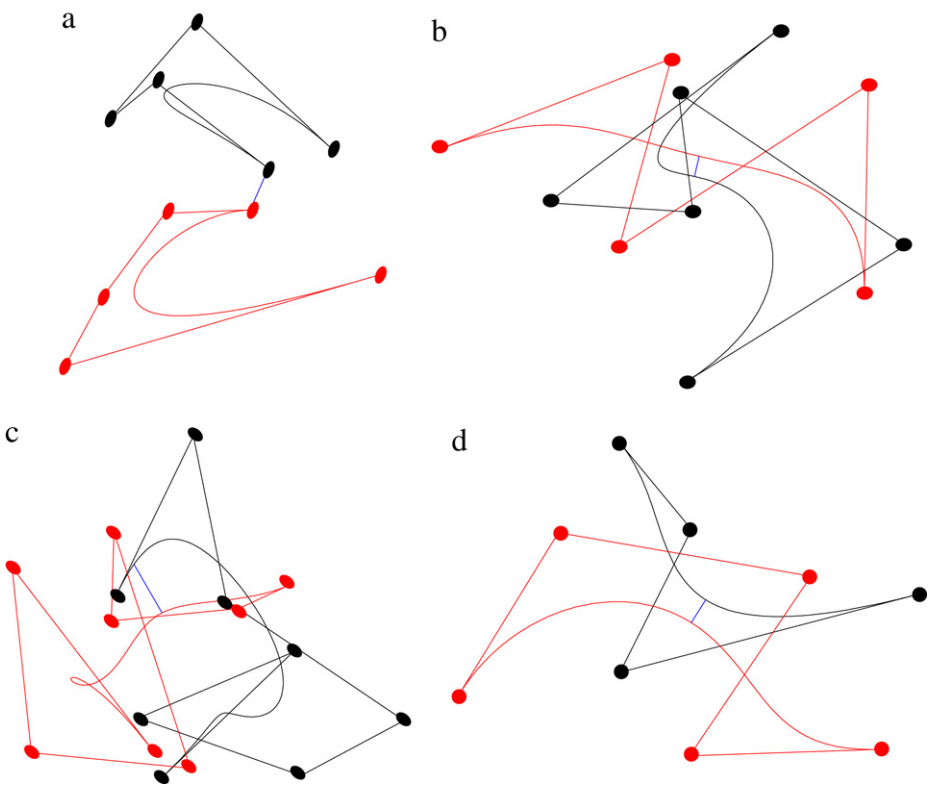

Fig. 3. Examples. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Comparisons with M_z and the new method M_n.

Case	Tol (δ)	M_z		M_n	
		T_s	T_c (ms)	T_s	T_c (ms)
Fig. 3(b)	10^{-5}	165	3.420	63	0.781
	10^{-6}	192	3.864	68	0.828
	10^{-8}	199	3.912	75	0.906
Fig. 3(c)	10^{-5}	338	15.043	31	1.250
	10^{-6}	405	17.670	39	1.485
	10^{-8}	468	20.964	47	1.656
Fig. 3(d)	10^{-5}	292	3.291	62	0.421
	10^{-6}	337	4.002	68	0.438
	10^{-8}	355	4.187	75	0.484

than subdivision time of the method in [10], and usually the total computation time of the new method is much less than that of the method in [10].

Fig. 3 shows four examples. The curves in red and in black are the two given curves, and the line segment in blue connects the two points in the nearest point pair. In Fig. 3(a), the nearest points on the two curves are end-points of the two curves and Property 1 is satisfied. In this case, there is no subdivision step in the new method and all the roots of the equation system (1) are unnecessary to solve. In Fig. 3(b–d), the nearest points are inner points of the two curves. The corresponding results are shown in Table 1. T_s and T_c denote the average subdivision time and the average computation time, M_z and M_n denote the method in [10] and the new method, respectively. The tolerance δ is the given resulting tolerance in the parameter domain, any two parameter may be considered the same as long as the absolute value of their difference is within δ. When the parameter interval size is within 10^{-4}, then the subdivision process is terminated in the method in [10]. As shown in Table 1, both the average subdivision time and the average computation time are less in the new method than that in the previous method in [10].

Fig. 4 shows such an example that the Newton method in the method in [10] may give a wrong answer. The Bézier curve in red has a cusp. The nearest points are framed by the rectangle in blue, which is amplified in Fig. 4(b). The new method leads to the line segment in black connecting the two nearest points. When the subdivision tolerance in the method in [10] is set as 10^{-3} or 10^{-4}, the Newton method leads to the line segment in red, which is wrong. When the subdivision tolerance

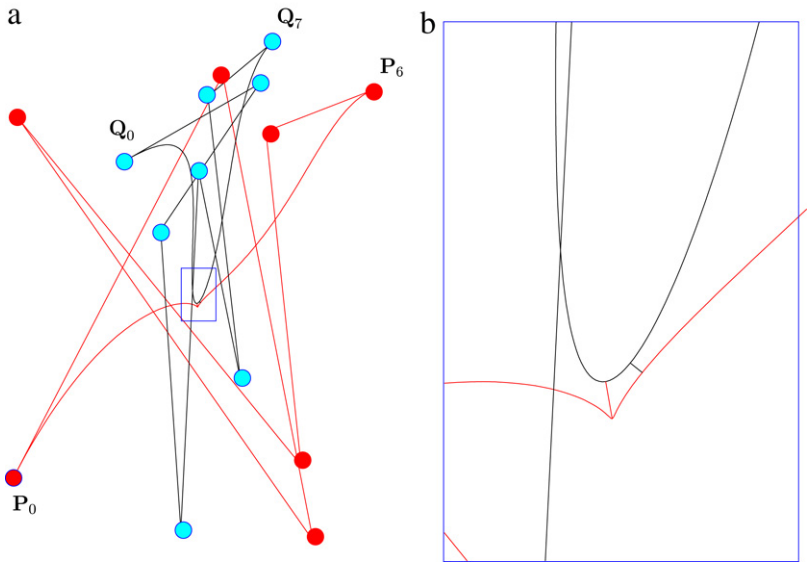

Fig. 4. An example which the Newton method may give a wrong answer. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

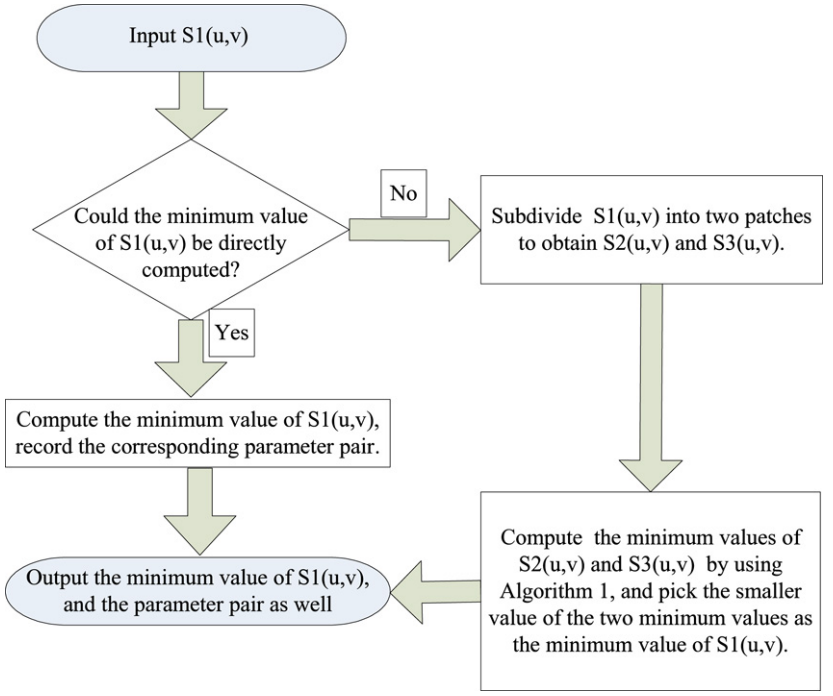

Fig. 5. The flow figure of Algorithm 1.

in the method in [10] is set as 10^{-5}, 10^{-6} or 10^{-7}, the Newton method could leads to the line in black, which is the same as that of the new method. All the examples are implemented in C++ with Windows PC, 1.7G CPU and 768M memory.

4. Conclusions

This paper presents a sweeping sphere clipping method for the minimum distance computation problem between two Bézier curves. A simple condition whether the nearest point is an end-point of a curve is provided. A heuristic method for the subdivision process is utilized to improve the efficiency of the new method. Compared with the root-finding method in [10], the new method can reduce most of the computation of the unnecessary solutions and need less computation time

on both Bézier form transformation and subdivision process. Examples are shown to illustrate the efficiency and robustness of the new method, which seems to be prior to the previous method in [10]. The future work will be to extend the new method to the curve–surface case and the surface–surface case.

Acknowledgements

The research was partially supported by Chinese 973 Program (2004CB719400), the National Science Foundation of China (60803076,60625202,60473130), INRIA, Chinese 863 Program (2007AA040401) and the Open Project Program of the State Key Lab of CAD&CG (A0804), Zhejiang University. The fifth author was supported by the Fok Ying Tung Education Foundation (111070). The authors would like to thank the referees for their helpful comments.

References

[1] X.D. Chen, H. Su, J.H. Yong, J.C. Paul, J.G. Sun, A counterexample on point inversion and projection for NURBS curve, Computer Aided Geometric Design 24 (2007) 302.
[2] X.D. Chen, Y. Zhou, Z. Shu, H. Su, J.C. Paul, Improved algebraic algorithm on point projection for Bézier curves, International Multi-Symposiums on Computer and Computational Sciences 17 (2007) 158–163.
[3] G. Elber, M. Kim, Geometric constraint solver using multivariate rational spline functions, in: Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications, 2001 pp. 1–10.
[4] S. Hu, J. Wallner, A second order algorithm for orthogonal projection onto curves and surfaces, Computer Aided Geometric Design 22 (2005) 251–260.
[5] D. Johnson, E. Cohen, A framework for efficient minimum distance computations, in: Proceedings of IEEE International Conference on Robotics & Automation, 1998, pp. 3678–3684.
[6] A. Limaiem, F. Trochu, Geometric algorithms for the intersection of curves and surfaces, Computer & Graphics 19 (1995) 391–403.
[7] M. Lin, D. Manocha, Fast interference detection between geometric models, The Visual Computer 11 (1995) 542–561.
[8] Y. Ma, W. Hewitt, Point inversion and projection for NURBS curve and surface: Control polygon approach, Computer Aided Geometric Design 20 (2003) 79–99.
[9] M. Mortenson, Geometric Modeling, Wiley, New York, 1985.
[10] N. Patrikalakis, T. Maekawa, Shape Interrogation for Computer Aided Design and Manufacturing, Springer-Verlag, New York, 2001.
[11] L. Piegl, W. Tiller, Parametrization for surface fitting in reverse engineering, Computer-Aided Design 33 (2001) 593–603.
[12] L. Piegl, W. Tiller, The NURBS Book, 2nd ed., Springer-Verlag, New York, 1997.
[13] J. Pegna, F. Wolter, Surface curve design by orthogonal projection of space curves onto free-form surfaces, Journal of Mechanical Design 118 (1996) 42–52.
[14] I. Selimovic, Improved algorithms for the projection of points on NURBS curves and surfaces, Computer Aided Geometric Design 23 (2006) 439–445.
[15] J.M. Zhou, E.C. Sherbrooke, N. Patrikalakis, Computation of stationary points of distance functions, Engineering with Computers 9 (1993) 231–246.