Computer Aided Geometric Design 29 (2012) 242–254

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Efficient point-projection to freeform curves and surfaces

Young-Taek Oh [a], Yong-Joon Kim [a], Jieun Lee [b,*], Myung-Soo Kim [a], Gershon Elber [c]

[a] School of Computer Science and Engineering, Seoul National Univ., Seoul 151-744, Republic of Korea
[b] School of Computer Engineering, Chosun University, Gwangju 501-759, Republic of Korea
[c] Computer Science Department, Technion, Haifa 32000, Israel

ARTICLE INFO

Article history:
Available online 22 April 2011

Keywords:
Point-projection
Nearest point
Minimum distance
Line/plane clipping
Circle/sphere clipping
Spiral curve
Evolute
Bisector curve
Voronoi cell

ABSTRACT

We present an efficient algorithm for projecting a given point to its closest point on a family of freeform curves and surfaces. The algorithm is based on an efficient culling technique that eliminates redundant curves and surfaces which obviously contain no projection from the given point. Based on this scheme, we can reduce the whole computation to considerably smaller subproblems, which are then solved using a numerical method. For monotone spiral planar curves with no inflection, we show that a few simple geometric tests are sufficient to guarantee the convergence of numerical methods to the closest point. In several experimental results, we demonstrate the effectiveness of the proposed approach.

1. Introduction

The projection of a point to a set is the closest element of the set to the given point. The point-projection problem plays a crucial role in many important geometric computations such as the Hausdorff distance computation, the minimum distance computation, simulation, haptic rendering, tolerance checking, freeform shape fitting and reconstruction, to mention only a few (Barton et al., 2010; Gilbert et al., 1988; Johnson, 2005; Kim et al., 2010; Lin and Canny, 1991; Lin and Gottschalk, 1998; Lin and Manocha, 2004; Tang et al., 2009). Because of its importance, many previous methods have been developed for the solution of the problem (Chen et al., 2008; Hu and Wallner, 2005; Johnson, 2005; Liu et al., 2009; Ma and Hewitt, 2003; Selimovic, 2006).

There are many variants of the point-projection problem, depending on what to project, where to project, which distance-metric to use, etc. In this paper, we consider a basic type of the point-projection problem, where we project a static point $\mathbf{p}$ to a finite family of freeform curves $C_i(t)$, $0 \leqslant t \leqslant 1$, or surfaces $S_i(u, v)$, $0 \leqslant u, v \leqslant 1$, for $i = 1, \ldots, n$. An immediate application of this problem can be found in an interactive selection of the nearest curve C_k to the cursor location $\mathbf{p}$ among all curves on the display screen. For a further manipulation of the selected curve C_k, we may also need to find the parameter t^* of the nearest curve point $C_k(t^*)$.

Translating the point $\mathbf{p}$ and the curves $C_i(t)$ or surfaces $S_i(u, v)$ by $-\mathbf{p}$, we may assume that the query point $\mathbf{p}$ is located at the origin. The problem is then reduced to finding an index k and a curve parameter t^* or surface parameters (u^*, v^*) such that

* Corresponding author.
E-mail address: jieunjadelee@gmail.com (J. Lee).

0167-8396/$ – see front matter
doi:10.1016/j.cagd.2011.04.002

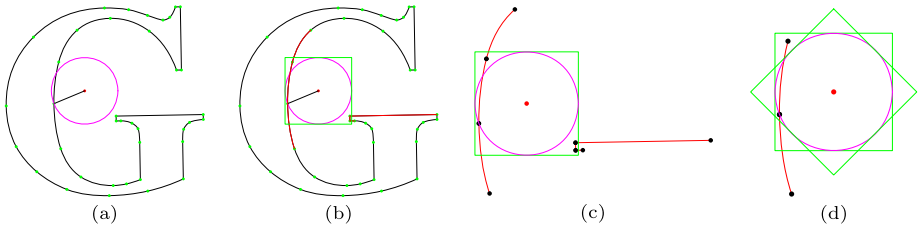

Fig. 1. Efficient culling via an axis-aligned bounding square for a clipping circle: (a) a clipping circle determined by the nearest sample point to the query point, (b) an axis-aligned bounding square for the clipping circle, (c) curves remaining after the culling stage, and (d) curves remaining after an additional culling with a rotated square.

$$\left\|C_k\left(t^*\right)\right\| = \min_{1\leqslant i\leqslant n}\ \min_{0\leqslant t\leqslant 1}\left\|C_i(t)\right\|, \quad \text{or} \quad \left\|S_k\left(u^*, v^*\right)\right\| = \min_{1\leqslant i\leqslant n}\ \min_{0\leqslant u,v\leqslant 1}\left\|S_i(u, v)\right\|.$$

Conventional approaches attack this problem for curves by solving the footpoint constraint: $\langle C_i(t), C_i'(t)\rangle = 0$, and then comparing the distances to all solution curve points of this equation as well as to the curve end points. For surfaces, the footpoint constraint is a system of equations: $\langle S_i(u, v), \frac{\partial S_i}{\partial u}(u, v)\rangle = 0$ and $\langle S_i(u, v), \frac{\partial S_i}{\partial v}(u, v)\rangle = 0$. The distances to all solution surface points are then compared with those to the four boundary curves of the surface. (See Johnson (2005) for an extensive literature survey on conventional methods.) Since the majority of these solutions turn out to be redundant, it is important to reduce the expensive equation solving as much as possible. Recent results propose efficient culling or clipping techniques for this purpose.

Chen et al. (2008) proposed a circle/sphere clipping method that applies a certain Bézier clipping technique to the squared distance function of a NURBS curve. This is an improvement over Selimovic (2006), which is based on a *Voronoi cell* test. The circle/sphere clipping demonstrates a better result. Nevertheless, as discussed in more details in Appendix A, it is quite expensive to set up the squared distance function which has degree almost twice higher than the original curve or surface degree(s). Moreover, it is also time-consuming to subdivide the squared distance function, in particular for the surface case.

In this paper, we propose an approach that is based on the clipping circle/sphere, but only conceptually; for efficiency reason, we instead use simple clipping lines/planes tangent to the circle/sphere. The simplest lines/planes for our purpose are those orthogonal to the coordinates axes. For example, given a clipping circle of radius r centered at the origin, we may consider the clipping lines $x = \pm r$ and $y = \pm r$. Fig. 1 shows an example where many input curves can easily be eliminated from further consideration using a simple window clipping. When a Bézier curve has control points (x_i, y_i) with $x_i > r$ for all i (or $x_i < -r$ for all i), the distance to this curve must be larger than r and can safely be eliminated. Similarly, we can check the y-coordinates of the control points. We may then proceed to checking $x_i + y_i > \sqrt{2}\, r$ for all i, etc. (Fig. 1(d)).

In some sense, our approach employs the k-DOP structure that bounds the control points of the freeform curves or surfaces (Klosowski et al., 1998). Though less tight than the convex hull of the control points, the k-DOP is considerably easier to construct; moreover, it is much tighter than the bounding circle/sphere of the control points. In our application, the k-DOP is constructed incrementally, which is essential for improving the overall performance of our algorithm. (According to our experiments, the first couple of separation tests usually cull away a large number of freeform curves and surfaces; thus the construction of a complete k-DOP is rarely needed.)

The real advantage of Chen et al. (2008) is in checking the uniqueness of local minimum of the squared distance function, which can be tested by the same condition for the control coefficients of the function. Nevertheless, it is difficult to extend the uniqueness condition of Chen et al. (2008) to the surface case. Consequently, Chen et al. (2009) propose no property for testing the uniqueness of local minimum distance between two freeform curves. A simple remedy for this deficiency is to employ the condition of Elber and Kim (2001) (Theorem 1) for testing the uniqueness of solution for a system of polynomial equations in a restricted domain.

Chen et al. (2008) pointed out that there may be redundant solutions in the system of equations that Elber and Kim (2001) solve. However, the chance of getting redundancy is extremely low in our approach. As the result of applying an efficient culling and clipping procedure, the remaining freeform curves and surfaces are usually defined on small domains. Consequently, the tests for the uniqueness of solution are mostly successful and no further expensive subdivisions are needed in the majority of cases. Once the uniqueness is guaranteed, the equations can be solved efficiently using a numerical method.

For the case of planar curves, Johnson (2005) proposed a curve segmentation approach in a preprocessing step so that one can guarantee the convergence of Newton's method to a unique local minimum distance point. In this paper, we show that a segmentation into monotone spiral curves (with no inflection) is sufficient for this purpose. We also develop a few simple geometric tests that can lead to an efficient algorithm for the point-projection problem.

The main contribution of this work can be summarized as follows:

- An efficient culling or clipping technique is proposed that can significantly reduce the problem size, i.e., the number of freeform curves and surfaces and also the size of their remaining subsegments and subpatches.

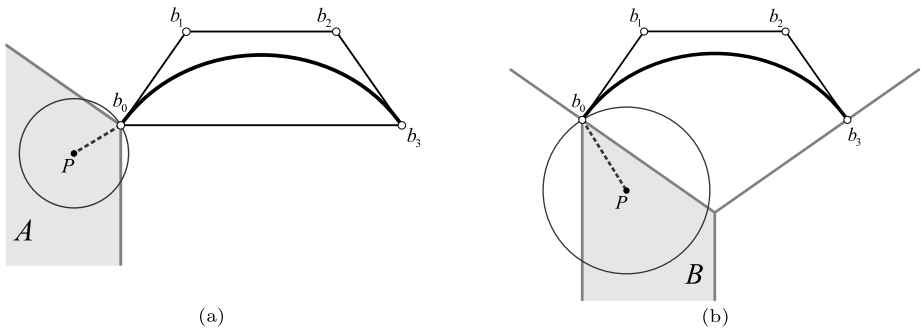

Fig. 2. (a) A region where each point is guaranteed to be projected to the left end point of the curve. (b) A similar region of points to be projected to $\mathbf{b}_0$, but which cannot be detected as such by the Voronoi cell test of the previous methods of Ma and Hewitt (2003) or Selimovic (2006).

- Our technique is based on geometric concepts such as the separating axis and the k-DOP bounding volume (Klosowski et al., 1998), though they are only incrementally constructed on-the-fly as needed for freeform curves and surfaces; thus our basic approach can easily be extended to more general projection problems or combined with other geometric computations based on these useful concepts.
- For the point-to-surface projection problem, employing the condition of Elber and Kim (2001) (Theorem 1), we can test the uniqueness of solution for a system of polynomial equations. This approach can be extended to more general multivariate projection problems.
- For monotone spiral curves with no inflection, we present a considerably more efficient algorithm that is based on the guaranteed convergence of numerical methods to the closest point. An important advantage of the geometry-based framework is that we can employ the circle-based numerical method of Hu and Wallner (2005) which has higher convergence rate than conventional Newton-type methods.

The rest of this paper is organized as follows. Section 2 presents a brief review on related previous work and also the basic idea of our approach. In Section 3, the details of our algorithm are presented. Section 4 considers spiral planar curves for which the point-projection problem can be greatly simplified. In Section 5, we demonstrate the effectiveness of our approach using several experimental results. Finally, in Section 6, we conclude this paper with discussions on future work.

2. Related work and our basic idea

In his PhD thesis, Johnson (2005) presented an extensive literature survey on the minimum distance computation and the associated applications in virtual prototyping and haptic rendering. He made an interesting observation that conventional methods for polygonal models mainly considered efficient pruning techniques; on the other hand, those for parametric surfaces developed numerical techniques for the minimum distance computation. Johnson (2005) then proposed a pruning technique for freeform curves and surfaces.

The method of Johnson (2005) starts with taking some samples on the freeform curves or surfaces and considers a circle/sphere centered at the query point and containing the nearest point among the samples. After that, the algorithm prunes away curves and surfaces when the convex hulls of their control points have no overlap with the current clipping circle/sphere. To do the overlap test, the distance is computed from the query point to each convex hull, which is the main computational bottleneck of the algorithm.

In this paper, we instead use separation axis tests with simple directions such as the coordinate directions and other combinations of them. This approach is essentially the same as incrementally constructing a k-DOP bounding volume for the set of all control points.

There are some previous methods for testing a sufficient condition for the point-projection to a curve end point or to a surface corner point. We start with discussing the method of Ma and Hewitt (2003). (Selimovic (2006) does the same thing but more efficiently; nevertheless, it is easier to visualize the condition of Ma and Hewitt (2003).) Fig. 2(a) shows a region A where each point $\mathbf{p}$ is guaranteed to be projected to the left end point $\mathbf{b}_0$ of a cubic Bézier curve. Consider the Voronoi cell decomposition for the convex hull of the Bézier control points. Then the region A is in fact the Voronoi cell for the end point $\mathbf{b}_0$ in the exterior of the convex hull; consequently, all points in the region A are closer to the end point $\mathbf{b}_0$ than to any other points in the convex hull including all the curve points. For higher degree curves, the convex hull computation can be somewhat intricate, in particular, for space curves. The same sufficient condition can be formulated for the surface case as well; however, the convex hull computation becomes even more cumbersome for the control points of a freeform surface.

Selimovic (2006) tests the same sufficient condition more efficiently. When a point $\mathbf{p}$ is in the Voronoi cell, it is on an outward normal line of the convex hull from the end point $\mathbf{b}_0$. Now consider a line/plane (passing through the end

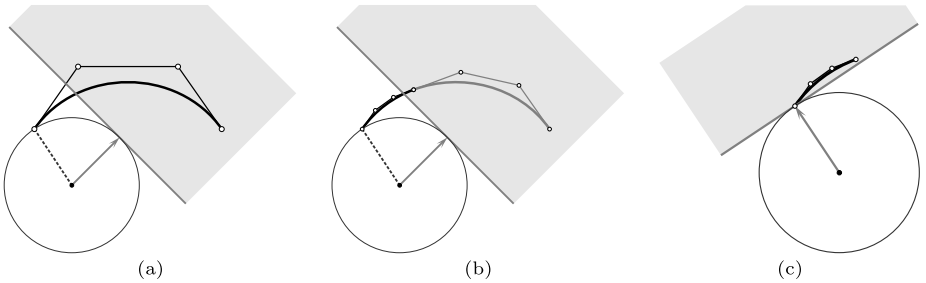

Fig. 3. Efficient curve clipping using tangent lines to the clipping circle: (a) a clipping line with normal $(1, 1)$, (b) a segment of the curve approximately clipped, and (c) the left end point is shown to be the closest point by the Voronoi cell condition of Selimovic (2006).

point $\mathbf{b}_0$) which is orthogonal to the normal line and divides the whole space into two half-spaces. All the Bézier control points are then contained in a half-space opposite to that of the query point $\mathbf{p}$. Selimovic (2006) checks this condition by testing $\langle \mathbf{b}_0 - \mathbf{p}, \mathbf{b}_i - \mathbf{b}_0 \rangle > 0$, for $i = 1, \ldots, d$. Since these sign tests are considerably easier than the construction of a convex hull and the Voronoi cell of a vertex, Selimovic (2006) is more efficient than Ma and Hewitt (2003).

Now the next question is whether the Voronoi cell concept is indeed a good idea for the point-projection problem. Fig. 2(b) shows a region B which may immediately lead us to a negative answer to this question. In this specific example of a symmetric cubic Bézier curve, all query points $\mathbf{p}$ in both regions A and B of Figs. 2(a) and 2(b) will be projected to the curve end point $\mathbf{b}_0$. Thus we need a better condition than the Voronoi cell test (Ma and Hewitt, 2003; Selimovic, 2006).

Geometrically speaking, the circle/sphere clipping of Chen et al. (2008) is optimal in the sense that, if a query point $\mathbf{p}$ has its projection to the end point $\mathbf{b}_0$, the circle with center $\mathbf{p}$ and radius $\|\mathbf{b}_0 - \mathbf{p}\|$ contains no other curve point in its interior. But the problem is how to test this condition efficiently. Chen et al. (2008) employed the squared distance function; however, as discussed in Appendix A, it is quite expensive to set up the squared distance function, in particular, for freeform surfaces and rational curves of high degree. Thus we propose more efficient geometric tests than the circle/sphere test.

Fig. 3(a) shows a clipping circle centered at $\mathbf{p}$ and a tangent line of the circle with normal $(1, 1)$. The Bézier clipping technique (Nishita et al., 1990; Sederberg and Nishita, 1990) applied to this line can remove some part of the curve as shown in Fig. 3(b). The remaining curve segment can be tested by the Voronoi cell condition of Selimovic (2006), which will guarantee the projection of $\mathbf{p}$ to $\mathbf{b}_0$ (Fig. 3(c)).

The problem becomes more difficult when the point $\mathbf{p}$ is located in the concave area of a curve and the projection occurs in the curve interior. For monotone spiral planar curves with no inflection, we have a relatively simple Voronoi cell structure for each curve element, which can be used for the design of an efficient point-projection algorithm (see Section 4). In the general case where the curvature monotonicity is not guaranteed a priori, the squared distance function of Chen et al. (2008) plays an important role as more details to be discussed in the following section.

3. Our approach

Our approach starts with sampling the given family of curves and surfaces, and making an initial guess of the minimum distance and the associated clipping circle/sphere. Simple tangent lines/planes are then considered for culling or clipping away redundant curves or surfaces. To the remaining curves and surfaces, geometric tests are applied to detect special cases of point-projection: (i) to a curve end point or a surface corner point, (ii) to a surface boundary curve, where the problem is reduced to a point-to-curve projection, or (iii) to a unique interior point of a curve segment or a surface patch, where a numerical method can be applied. Otherwise, the freeform curves and surfaces are further subdivided and the whole procedure is repeated recursively.

3.1. Clipping circle/sphere and clipping lines/planes

Given a query point $\mathbf{p}$ and its nearest point $\mathbf{p}_k$ among all sample points $\mathbf{p}_i$, $i = 1, \ldots, n$, we consider the circle/sphere with center $\mathbf{p}$ and radius $r = \|\mathbf{p}_k - \mathbf{p}\|$ that will be used for culling or clipping redundancies from the given freeform curves or surfaces. To make the whole algorithm efficient, we should take only a suitable number of sample points that would be sufficient to give a good initial guess on the minimum distance. When there are many curve segments or surface patches to consider, we take only the curve end points and the surface corner points into account at the beginning. As we converge to a small number of freeform curves and surfaces, we may take more samples in their interior.

For the sake of simplicity of presentation, we consider the curve case in detail; however, a similar technique can be applied to surfaces as well. Given a curve $C(t)$, the clipping based on a quadratic distance: $\|C(t) - \mathbf{p}\|^2 > r^2$, is quite expensive as discussed in Appendix A. Thus we instead do an approximate clipping with a few simple tangent lines to the clipping circle: $ax + by + c > 0$, where $c^2 = (a^2 + b^2)r^2$ and (a, b) is an outward normal direction of the tangent line. When all the control points $\mathbf{b}_i = (x_i, y_i)$, $i = 0, \ldots, d$, satisfy the following condition:

$$ax_i + by_i + c > 0, \quad \text{for } i = 0, \ldots, d,$$

we can guarantee that the whole curve $C(t)$ is outside the clipping circle as well as the clipping line and thus contains no projection from the query point $\mathbf{p}$. However, this approach requires $2(d+1)$ multiplications, $2(d+1)$ additions, and $(d+1)$ sign tests. We can do better. Following the approach of k-DOP (Klosowski et al., 1998), we consider simple normal directions such as $(\pm 1, 0)$, $(0, \pm 1)$, $(\pm 1, \pm 1)$, etc. Using these direction vectors, many multiplications in the culling/clipping tests can be replaced by additions and subtractions of x_i's and y_i's. In the majority of cases, only one or two of these directions would be needed to cull away redundancies.

3.2. Uniqueness of solution

After the culling and clipping stage, we will end up with a relatively small number of freeform curve segments or surface patches. For each of these remaining curves and surfaces, we compute the nearest point from the query point $\mathbf{p}$ and dynamically update the minimum distance and the clipping circle/sphere when a closer projection point is found than the current one.

We consider a Bézier curve $C(t)$ of degree d defined by $(d+1)$ control points $\mathbf{b}_i$, for $i = 0, \ldots, d$. If $\mathbf{p}$ is closer to $\mathbf{b}_0$ than to $\mathbf{b}_d$, we test the projection of $\mathbf{p}$ to the end point $\mathbf{b}_0$ using the Voronoi cell condition of Selimovic (2006):

$$\langle \mathbf{b}_0 - \mathbf{p}, \mathbf{b}_i - \mathbf{b}_0 \rangle > 0, \quad \text{for } i = 1, \ldots, d.$$

If the query point $\mathbf{p}$ is closer to $\mathbf{b}_d$ than to $\mathbf{b}_0$, the projection of $\mathbf{p}$ to the other end point $\mathbf{b}_d$ is tested as follows:

$$\langle \mathbf{b}_d - \mathbf{p}, \mathbf{b}_i - \mathbf{b}_d \rangle > 0, \quad \text{for } i = 0, \ldots, d-1.$$

Even if the above Voronoi cell condition is not met, we cannot completely exclude the possibility of projection to $\mathbf{b}_0$ or $\mathbf{b}_d$ (as shown by the region B of Fig. 2(b)); however, the chance is quite low since the region B is very small for a short curve segment we are dealing with after the culling and clipping stage. Thus we employ the squared distance function: $\|C(t) - \mathbf{p}\|^2$ of Chen et al. (2008). When the Bézier control coefficients f_i, $i = 0, \ldots, 2d$, of this function have only one local minimum, the function graph will have a U-shape with only one local minimum, which is thus the global minimum. (The exact curve location for the minimum distance can be computed by a numerical method.) Otherwise, the squared distance function is subdivided into two and each subproblem is tested recursively.

For a Bézier surface $S(u, v)$ of degree (d_1, d_2) defined by $(d_1+1)(d_2+1)$ control points $\mathbf{b}_{ij}$, for $i = 0, \ldots, d_1$ and $j = 0, \ldots, d_2$, we may assume that $\mathbf{p}$ is closer to $\mathbf{b}_{00}$ than to three other corner points. We can then test the projection of $\mathbf{p}$ to the surface corner point $\mathbf{b}_{00}$ using the condition of Selimovic (2006):

$$\langle \mathbf{b}_{00} - \mathbf{p}, \mathbf{b}_{ij} - \mathbf{b}_{00} \rangle > 0, \quad \text{for } i = 0, \ldots, d_1,\ j = 0, \ldots, d_2,\ i + j > 0.$$

Selimovic (2006) also presents a condition for checking the projection of $\mathbf{p}$ to a boundary curve $S(u, 0)$:

$$\left\langle \mathbf{b}_{i0} - \mathbf{p}, \frac{\partial S}{\partial v}(u, v) \right\rangle > 0, \quad \text{for } i = 0, \ldots, d_1, \text{ and } 0 \leqslant u, v \leqslant 1.$$

The projection to other boundary curves can be tested similarly.

However, the above condition is difficult to check since the set of all v-partial derivatives forms another freeform surface of degree $(d_1, d_2 - 1)$. From the relation: $\frac{\partial S}{\partial v}(u, v) = \sum_{l=0}^{d_1} \sum_{j=0}^{d_2-1} (\mathbf{b}_{l,j+1} - \mathbf{b}_{l,j}) B_l^{d_1}(u) B_j^{d_2-1}(v)$, we suggest a simpler sufficient condition for the projection of $\mathbf{p}$ to the boundary curve $S(u, 0)$:

$$\langle \mathbf{b}_{i,0} - \mathbf{p}, \mathbf{b}_{l,j+1} - \mathbf{b}_{l,j} \rangle > 0, \quad \text{for all } i, l = 0, \ldots, d_1,\ j = 0, \ldots, d_2 - 1.$$

Now when the above conditions are not met, the minimum distance may occur in the surface interior (even though we cannot exclude some possibility of getting the minimum distance on the boundary curve or even at a corner point). Using the condition of Elber and Kim (2001) (Theorem 1), we consider how to test the uniqueness of local minimum distance on the interior of the surface $S(u, v)$: $0 < u, v < 1$. At each local minimum distance, the solution point $S(u, v)$ must satisfy the following two bivariate equations:

$$F(u, v) = \left\langle S(u, v), \frac{\partial S}{\partial u}(u, v) \right\rangle = 0, \qquad G(u, v) = \left\langle S(u, v), \frac{\partial S}{\partial v}(u, v) \right\rangle = 0.$$

When there is a unique solution for this system of equations in the uv-domain: $0 < u, v < 1$, this solution may correspond to the minimum distance from the query point $\mathbf{p}$.

The above system of bivariate equations has at most one solution if the following condition is met (Elber and Kim, 2001):

$$\left\{ \alpha \nabla F(u, v) \;\middle|\; \alpha \in R, 0 < u, v < 1 \right\} \cap \left\{ \beta \nabla G(u, v) \;\middle|\; \beta \in R, 0 < u, v < 1 \right\} = \{\mathbf{0}\}.$$

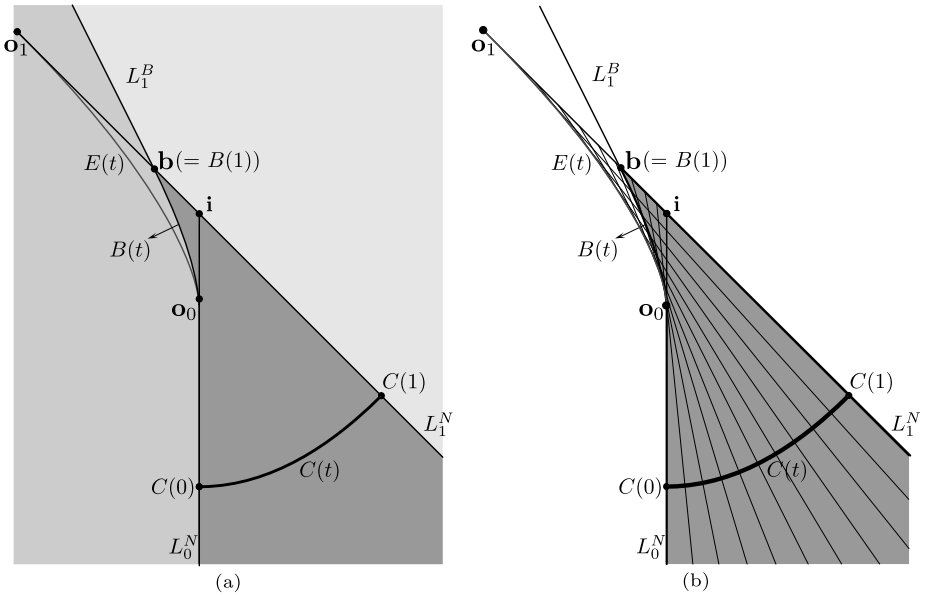

Fig. 4. A spiral curve and its Voronoi diagram: (a) the regions of different shades correspond to the Voronoi cells of $C(0)$, $C(1)$, and the curve interior, and (b) the covering number implies the number of footpoint(s).

3.3. Numerical improvement

For the curve case, we compute the minimum of the squared distance function $D^2(t) = \sum_{i=0}^{2d} f_i B_i^{2d}(t)$, by solving the unique solution of the following derivative equation:

$$\sum_{i=0}^{2d-1} (f_{i+1} - f_i) B_i^{2d-1}(t) = 0.$$

We employ Brent's method for the sake of robustness in solving the above equation (Press et al., 2007). Note that the uniqueness of solution is guaranteed by the condition of Chen et al. (2008).

For the surface case, we employ a bivariate Newton–Raphson method as discussed in Elber and Kim (2001) and implemented in the IRIT solid modeling system (IRIT, 2005).

4. Acceleration technique for spiral planar curves

In this section, we consider a special type of planar curves for which the point-projection problem can be greatly simplified. For these curves, a numerical search for a footpoint is sufficient for detecting the closest point. The monotonicity of curvature plays an important role here. A regular planar curve $C(t)$, $0 \leqslant t \leqslant 1$, is called *spiral* if its signed curvature $\kappa(t)$ is continuous and strictly monotone (Barton and Elber, 2011). In this paper, we also assume that each spiral curve has no inflection in its interior and is monotone along the x- and y-directions.

For a plane curve $C(t)$, its evolute $E(t) = C(t) + \frac{1}{\kappa(t)} N(t)$ is generated by the centers of osculating circles of $C(t)$, where $N(t)$ is the unit normal of $C(t)$. The evolute can also be generated as an envelope of the normal lines of $C(t)$ (see p. 23 of do Carmo (1976)). For a spiral curve $C(t)$, its evolute $E(t)$ has no cusp. Each tangent line from a query point $\mathbf{p}$ to the evolute $E(t)$ is the normal line of $C(t)$ (do Carmo, 1976). If the point $\mathbf{p}$ is on the normal line segment $\overline{E(t)C(t)}$, $\mathbf{p}$ draws a footpoint to the curve location $C(t)$. But the footpoint $C(t)$ may not be the closest point to $\mathbf{p}$ since the end point $C(0)$ is closer when $\mathbf{p}$ is in the Voronoi cell of $C(0)$.

Fig. 4(a) shows a spiral curve $C(t)$ and its evolute $E(t)$. Also shown are the bisector curve $B(t)$ defined by $C(0)$ and the curve interior $C(t)$, $0 < t < 1$, and the bisector line L_1^B of $C(0)$ and $C(1)$. The Voronoi cells for $C(0)$, $C(1)$, and the curve interior $C(t)$, $0 < t < 1$, are shown in different shades. The osculating circle of $C(t)$ has its center at $\mathbf{o}_t$, which traces the evolute $E(t) = \mathbf{o}_t$, $0 \leqslant t \leqslant 1$. The bisector curve $B(t)$, $0 \leqslant t \leqslant 1$, starts from $\mathbf{o}_0$ and ends at $\mathbf{b}$ $(= B(1))$. Note that the point $\mathbf{b}$ is the intersection between the bisector line L_1^B and the normal line L_1^N. This is because there are two equivalent ways of generating the bisector curve $B(t)$ (see pp. 203–204 of Peternell (2000)): (i) an envelope of bisector lines L_t^B between $C(0)$ and $C(t)$, and (ii) a set of line–line intersections $\{L_t^B \cap L_t^N \mid 0 < t \leqslant 1\}$.

Fig. 4(b) shows a family of normal lines and the evolute $E(t)$ as their envelope. The region A (bounded by $E(t)$, $\overline{\mathbf{o}_0\mathbf{i}}$, and $\overline{\mathbf{i}\mathbf{o}_1}$) has a double covering of normal lines. On the other hand, the region B (bounded by two normal lines L_0^N and L_1^N

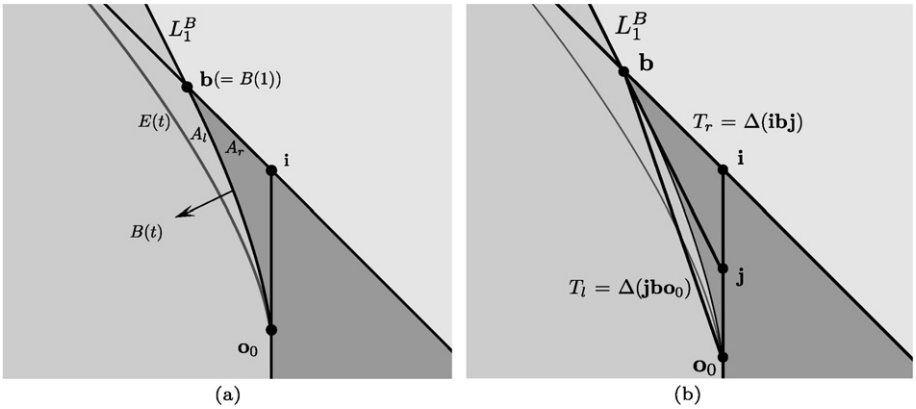

Fig. 5. Close-up views of regions around $E(t)$ and $B(t)$: (a) the bisector curve $B(t)$ is in between the Voronoi cells of $C(0)$ and the curve interior, and (b) the bisector line L_1^B is in between the Voronoi cells of $C(0)$ and $C(1)$.

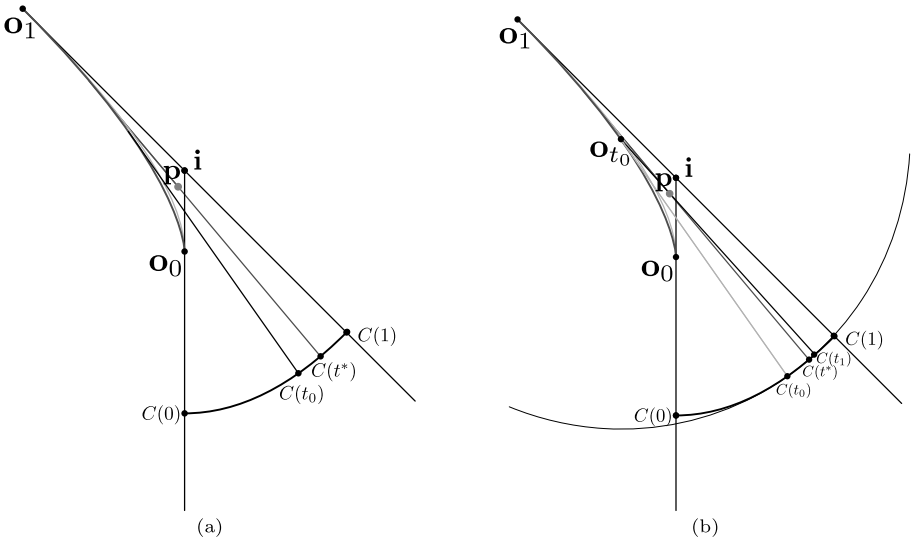

Fig. 6. Footpoint computation: (a) $C(t^*)$ is the exact footpoint of $\mathbf{p}$ and an initial solution t_0 is taken as the larger parameter $t_0 = \beta\ (> \alpha) \in (0, 1)$, where one can draw tangent line from $\mathbf{p}$ to $Q(t)$, a quadratic approximation curve to the evolute $E(t)$, and (b) the next step $C(t_{i+1})$ is computed using the projection of $\mathbf{p}$ to the osculating circle at $C(t_i)$ (Hu and Wallner, 2005).

meeting at $\mathbf{i}$) has only one covering. The number of covering corresponds to the number of footpoint(s) one can draw from a query point $\mathbf{p}$ to the curve $C(t)$. When $\mathbf{p}$ is inside B (with one covering), there will be only one footpoint from $\mathbf{p}$ to the curve $C(t)$, which is also the closest point.

Fig. 5(a) shows a close-up view of the region A which is divided by the bisector curve $B(t)$ into two subregions A_l and A_r. The subregion A_l belongs to the Voronoi cell of $C(0)$, whereas the subregion A_r is contained in the Voronoi cell of the curve interior. A close-up view of A_r is given in Fig. 5(b), where the bisector line L_1^B is extended to meet the normal line L_0^N at $\mathbf{j}$. When $\mathbf{p} \in T_r\ (= \Delta(\mathbf{ibj}))$, the point $\mathbf{p}$ is in the Voronoi cell of the curve interior and thus $\mathbf{p}$ has its closest point at $C(t)$, for some $0 < t \leqslant 1$. In this case, $\mathbf{p}$ has two local footpoints $C(\alpha)$ and $C(\beta)$ $(0 < \alpha < \beta \leqslant 1)$ and the footpoint $C(\beta)$ $(\beta > \alpha)$ is the closest point. A simple way to detect the larger parameter β is to start a numerical search with an initial guess $\beta = 1$. When $\mathbf{p} \in T_l = \Delta(\mathbf{jbo}_0)$, which is the convex hull of the bisector curve $B(t)$, the point $\mathbf{p}$ can be either in the Voronoi cell of $C(0)$ or in the Voronoi cell of the curve interior. Thus there is no guarantee that the computed local footpoint will be the closest point to the curve. The result of footpoint computation should be compared with the distance to the end point $C(0)$.

Now we discuss how to start a numerical iteration for the footpoint computation. Drawing tangent lines to the exact evolute $E(t)$ can be expensive since $E(t)$ is a rational curve of degree $3d - 3$ when $C(t)$ is a polynomial curve of degree d. Thus we employ an approximation of $E(t)$ by a quadratic Bézier curve $Q(t)$ with control points $\mathbf{o}_0$, $\mathbf{i}$, and $\mathbf{o}_1$. Tangent lines from $\mathbf{p}$ to $Q(t)$ can be determined by closed-form solution(s) of a quadratic equation. There are at most two real roots and we take the larger one $t_0 = \beta\ (> \alpha) \in (0, 1)$ as an initial solution. Fig. 6(a) shows an approximate solution $C(t_0)$ thus

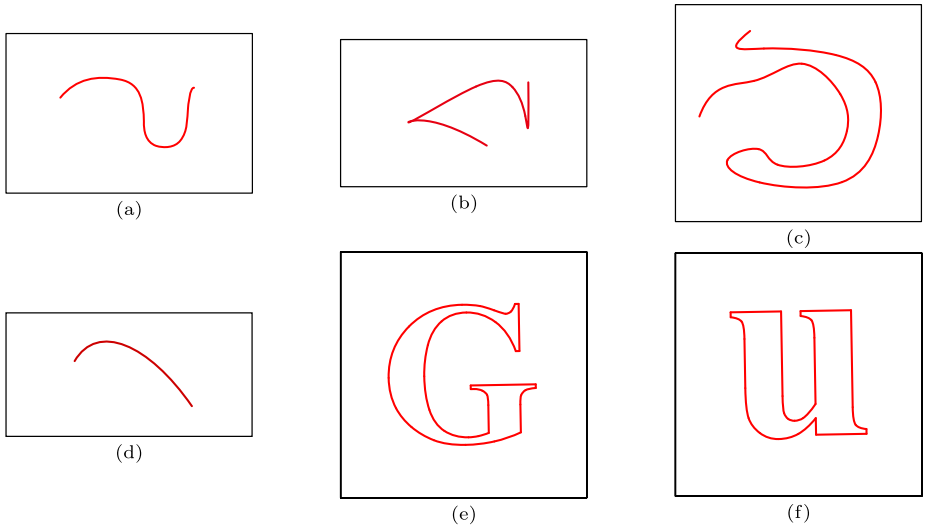

Fig. 7. Planar curves composed of: (a)–(d) cubic Bézier curve segments, and (e)–(f) linear and quadratic Bézier curve segments.

computed. In general the query point $\mathbf{p}$ is not located on the normal line $L^N_{t_0}$ though it should be on the normal line $N^N_{t^*}$ at its exact footpoint $C(t^*)$. In Fig. 6(b), employing the point-projection algorithm of Hu and Wallner (2005), we project the query point $\mathbf{p}$ to the osculating circle at $C(t_0)$, and update the search to another location $C(t_1)$, and then repeat the same procedure.

5. Experimental results

We have implemented our point-projection algorithm in C on an Intel Pentium IV 2.4 GHz PC with a 2 GB main memory. To demonstrate the effectiveness of our approach, we have tested the algorithm for several freeform curves and surfaces.

Fig. 7 shows six examples of planar curves, each consisting of 6, 5, 15, 1, 38, and 27 Bézier curve segments, respectively. (The first four examples include cubic Bézier curves only and the last two examples are composed of linear and quadratic Bézier curves.) In each example, the end points of Bézier curves are used as samples for estimating the initial guess on the minimum distance. To compare the performance of different algorithms, we randomly generate 100 query points within a box that is 50% larger in each dimension than the axis-aligned minimum bounding box of the given curve. Table 1 shows the result of measuring the performance of different algorithms on these query points averaged over 100 independent tests.

For each random query point $\mathbf{p}$, the distances to the Bézier curve end points are compared and the minimum is taken as the radius of an initial clipping circle centered at the query point $\mathbf{p}$. This part is common in all algorithms under comparison and thus not included in the performance measure.

In Table 1, the column under D^2 ONLY shows the result of Chen et al. (2008) applied to the Bézier curves. The column under CIRCLE + D^2 corresponds to an algorithm that first culls away some redundant Bézier curves when their bounding circles have no overlap with the initial clipping circle. (The bounding circle of a Bézier curve has its center at the center of mass of the Bézier control points and its radius is taken to be the maximum distance from the center to the control points.) After that, the algorithm employs Chen et al. (2008) for the remaining Bézier curves. The next column under KDOP + D^2 shows the result of applying a k-DOP based culling to the Bézier curves and then employing Chen et al. (2008) for the remaining Bézier curves. For the k-DOP based tests, we have employed only a subset of 8 directions: $(\pm 1, 0)$, $(0, \pm 1)$, $(\pm 1, \pm 1)$. The last column under K + S + D^2 reports the result from a variant of our algorithm where we apply Selimovic (2006) immediately after the k-DOP based culling and right before we employ Chen et al. (2008).

The first four rows in each of Table 1(a)–(f) show the computing time for each algorithm in each step of the computation, measured in microseconds (μs). The first row reports the time taken in culling or clipping redundant Bézier curves. The second row shows the computing time for setting up the squared distance functions for all remaining Bézier curves and checking the uniqueness of local minimum for each function and recursively subdividing those with potentially multiple local minimums. The third row is for the stage of numerical improvement. The total computing time is shown in the fourth row.

The last three rows in each of Table 1(a)–(f) show the average number of Bézier curves remaining after the initial culling, the number of subdivisions taken in all the squared distance functions $D^2(t)$ until the uniqueness of local minimum is guaranteed for each curve segment, and the total number of Brent iterations for all subdivided curves in the numerical improvement stage. The numerical approximation is made within a precision of 10^{-6} in the Bézier curve parameter. All the data in each table show the average of results over 100 independent tests on randomly selected query points.

Table 1
Results for the six planar curves in Fig. 7.

	D^2 ONLY	CIRCLE + D^2	KDOP + D^2	K + S + D^2
(a)				
Cull./clip.	0.000	24.180	25.030	23.970
D^2 + subdiv.	122.110	32.550	24.020	21.400
Numeric	9.890	5.490	4.850	5.330
Total (in μs)	132.000	62.220	53.900	50.700
# curve	6.000	1.960	1.490	1.350
# subdiv.	0.120	0.100	0.080	0.080
# num. iter.	10.000	5.940	4.980	4.980
(b)				
Cull./clip.	0.000	22.350	20.260	20.930
D^2 + subdiv.	110.120	26.240	20.380	20.480
Numeric	10.380	5.140	4.610	4.600
Total (in μs)	120.500	53.730	45.250	46.010
# curve	5.000	1.600	1.280	1.230
# subdiv.	0.880	0.500	0.320	0.320
# num. iter.	11.860	5.780	5.300	5.040
(c)				
Cull./clip.	0.000	54.940	50.160	50.220
D^2 + subdiv.	291.570	30.430	26.600	21.340
Numeric	21.140	5.060	4.610	4.490
Total (in μs)	312.710	90.430	81.370	76.050
# curve	15.000	2.030	1.730	1.420
# subdiv.	0.700	0.140	0.070	0.070
# num. iter.	23.600	5.060	4.480	4.480
(d)				
Cull./clip.	0.000	7.160	6.890	6.860
D^2 + subdiv.	23.930	19.280	16.580	10.280
Numeric	3.020	3.120	3.110	2.130
Total (in μs)	26.950	29.560	26.580	19.270
# curve	1.000	1.000	1.000	0.570
# subdiv.	0.270	0.270	0.270	0.270
# num. iter.	2.530	2.530	2.530	1.620
(e)				
Cull./clip.	0.000	111.623	112.147	112.607
D^2 + subdiv.	553.575	37.542	27.580	23.069
Numeric	20.079	6.856	5.637	5.707
Total (in μs)	573.655	156.022	145.364	141.384
# curve	38.000	2.690	1.980	1.580
# subdiv.	0.010	0.000	0.000	0.000
# num. iter.	20.480	5.260	4.140	4.140
(f)				
Cull./clip.	0.000	81.500	76.910	78.650
D^2 + subdiv.	440.020	35.500	25.500	20.520
Numeric	16.000	5.100	3.820	3.870
Total (in μs)	456.020	122.100	106.230	103.040
# curve	27.000	2.780	2.080	1.500
# subdiv.	0.070	0.010	0.000	0.000
# num. iter.	13.620	3.990	2.750	2.470

Note that, in the first column D^2 ONLY of Table 1(e)–(f), the number of Brent iterations is smaller than the number of remaining curve segments. The remaining segments are the same as the original curve segments since there is no initial culling step in this method. This is because, for some curve segments, the control coefficients of the squared function $D^2(t)$ are all larger than the current minimum squared distance and thus no numerical iteration is needed for these redundant curves.

In Table 1, we can observe that the computing times for culling redundant curve segments are about the same in the two different methods: CIRCLE + D^2 and KDOP + D^2. However, the latter is more effective in the culling result itself, since the k-DOP bounding volume is usually tighter than the circle/sphere bounding volume. In Tables 1(a) and 1(c)–(f), some performance improvement is achieved by employing the end-point projection condition of Selimovic (2006) as reported in the last column under K + S + D^2, though it is not obvious in Table 1(b).

Table 2
Results for the same planar curves (Fig. 7) but segmented to monotone spiral curves with no inflection.

	D^2 only	Circle + D^2	Kdop + D^2	K + S + D^2	Evolute
(a)					
Cull./clip.	0.000	13.420	13.450	14.760	0.000
D^2 + subdiv.	123.790	27.210	21.410	15.510	0.000
Numeric	8.790	4.580	4.250	3.210	7.670
Total (in µs)	132.580	45.210	39.110	33.480	7.670
# curve	12.000	2.330	1.760	1.300	12.000
# subdiv.	0.070	0.060	0.050	0.040	0.000
# num. iter.	9.080	4.560	4.100	3.080	3.880
(b)					
Cull./clip.	0.000	18.460	17.500	19.280	0.000
D^2 + subdiv.	158.710	27.270	21.540	21.400	0.000
Numeric	12.060	4.960	4.070	2.830	9.970
Total (in µs)	170.770	50.690	43.110	43.510	9.970
# curve	16.000	2.400	1.830	1.360	16.000
# subdiv.	0.080	0.030	0.000	0.000	0.000
# num. iter.	13.770	5.480	4.020	2.560	5.690
(c)					
Cull./clip.	0.000	46.700	44.280	45.770	0.000
D^2 + subdiv.	404.470	24.940	24.440	22.430	0.000
Numeric	19.940	4.030	3.900	3.480	17.320
Total (in µs)	424.410	75.670	72.620	71.680	17.320
# curve	43.000	2.210	2.180	2.030	43.000
# subdiv.	0.120	0.000	0.000	0.000	0.000
# num. iter.	22.260	4.010	3.840	3.490	9.370
(d)					
Cull./clip.	0.000	4.720	3.910	4.750	0.000
D^2 + subdiv.	33.210	13.220	13.530	13.310	0.000
Numeric	3.760	1.470	1.480	1.470	2.760
Total (in µs)	36.970	19.410	18.920	19.530	2.760
# curve	3.000	1.120	1.120	1.120	3.000
# subdiv.	0.090	0.090	0.090	0.090	0.000
# num. iter.	3.350	1.040	1.040	1.040	1.030
(e)					
Cull./clip.	0.000	91.480	84.620	87.160	0.000
D^2 + subdiv.	465.740	18.560	15.020	12.810	0.000
Numeric	15.790	3.930	3.530	2.980	18.110
Total (in µs)	481.530	113.970	103.170	102.950	18.110
# curve	84.000	6.210	3.280	2.820	84.000
# subdiv.	0.010	0.000	0.000	0.000	0.000
# num. iter.	17.140	3.940	3.430	2.630	14.340
(f)					
Cull./clip.	0.000	52.570	51.280	54.420	0.000
D^2 + subdiv.	226.110	13.620	12.730	12.070	0.000
Numeric	8.210	2.160	2.160	1.840	5.600
Total (in µs)	234.320	68.350	66.170	68.330	5.600
# curve	49.000	5.540	3.970	3.760	49.000
# subdiv.	0.030	0.000	0.000	0.000	0.000
# num. iter.	9.100	1.700	1.660	1.280	2.820

In Table 2, we repeat the same algorithms to the same six examples of planar curves but segmented to monotone spiral curves with no inflection point in a preprocessing step. We compare their performance with our point-projection algorithm specialized for spiral curve segments as discussed in Section 4. As expected, the performance of our evolute-based algorithm (shown in the last column under Evolute is considerably better than others). On the other hand, we can observe that the number of curve segments increases about twice or three times.

Fig. 8 shows two examples of space curves. In the space case, for the k-DOP tests, we have employed only a subset of 14 directions: $(\pm1, 0, 0)$, $(0, \pm1, 0)$, $(0, 0, \pm1)$, $(\pm1, \pm1, \pm1)$. Their performance results are shown in Table 3.

Fig. 9(a) shows the spout of the Utah teapot represented as a non-uniform bicubic B-spline surface which can be converted to 4 bicubic Bézier surfaces. Fig. 9(b) shows the B-spline surfaces for the whole Utah teapot which can be converted to 28 bicubic Bézier surfaces. In the surface case, after culling away redundant Bézier surfaces, we compute the local mini-

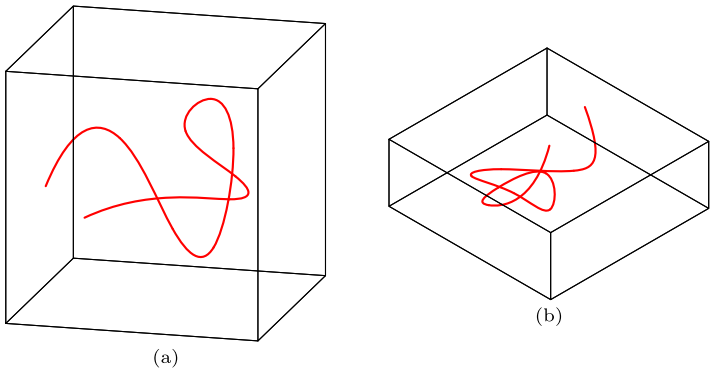

Fig. 8. Space curves composed of cubic Bézier curve segments.

Table 3
Results for two space curves in Fig. 8.

	D^2 ONLY	SPHERE + D^2	KDOP + D^2	K + S + D^2
(a)				
Cull./clip.	0.000	30.687	30.913	31.324
D^2 + subdiv.	145.569	36.451	23.657	22.148
Numeric	19.949	8.167	5.673	5.582
Total (in μs)	165.518	75.306	60.243	59.054
# curve	9.000	2.630	1.700	1.570
# subdiv.	0.610	0.250	0.120	0.120
# num. iter.	24.400	9.030	6.070	5.920
(b)				
Cull./clip.	0.000	33.430	30.880	32.690
D^2 + subdiv.	163.310	27.070	20.010	19.390
Numeric	16.240	3.800	3.150	2.870
Total (in μs)	179.550	64.300	54.040	54.950
# curve	9.000	1.710	1.230	1.210
# subdiv.	0.670	0.210	0.040	0.040
# num. iter.	18.080	4.150	2.670	2.670

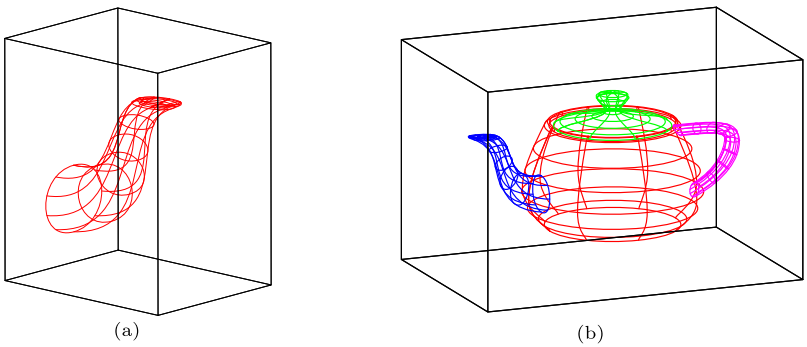

Fig. 9. The B-spline surfaces for (a) the spout and (b) the Utah teapot.

mum distance to each of the remaining Bézier surfaces by employing the bivariate equation solver of Elber and Kim (2001) as implemented in the IRIT solid modeling system (IRIT, 2005).

In Table 4, the first row shows the conversion time from the B-spline representation to bicubic Bézier surfaces, the second row shows the culling time, and the third row shows the time taken in the solution procedure by the IRIT solver. Because of the large number of control points for the surface case, we can observe that the sphere method takes more culling time than the k-DOP method. Similarly to the curve case, for the culling result itself, the k-DOP is more effective than the sphere.

Table 4
Results for (a) the spout (Fig. 9(a)), and (b) the Utah teapot (Fig. 9(b)).

	Sphere + D^2	Kdop + D^2
(a)		
Conversion	43.023	43.064
Culling	9.897	7.516
IRIT	1186.613	1005.708
Total (in μs)	1196.510	1013.224
# surface	2.220	1.890
(b)		
Conversion	438.731	434.625
Culling	35.287	25.906
IRIT	1690.218	1228.548
Total (in μs)	1725.505	1254.455
# surface	5.420	3.810

6. Conclusions

We have presented an efficient point-to-curve/surface projection algorithm that computes the nearest point on a family of freeform curves and surfaces from a given query point. Effectively using only a small number (usually one or two) of separation axes among the k-DOP directions, we have developed a culling method as efficient as the circle/sphere method for the curve case and even more effective than the sphere for the surface case. Tighter than circles/spheres, our approach produces better culling results and consequently better performance for both curve and surface cases.

For a special type of planar curves, i.e., monotone spiral curves with no inflection, we have introduced a few simple geometric tests that can guarantee the detection of the closest point from a footpoint computation. Combined with the algorithm of Hu and Wallner (2005), we have developed a considerably more efficient point-projection algorithm than others.

In future work, we plan to extend the current result to the case of projecting a dynamically moving point to freeform curves and surfaces. In this more general case, we believe that a pre-processing of the freeform shapes into simple ones such as spiral curves would be quite useful since we can fully utilize their geometric structure. Furthermore, we hope our approach could be extended to the more general distance problems dealing with freeform shapes under continuous deformation.

Acknowledgements

This research was supported in part by the Israeli Ministry of Science Grant No. 3-4642, and in part by NRF Research Grants (No. 2010-0014351 and No. 2010-0005597). Y.-T. Oh was supported by the Seoul Fellowship.

Appendix A. Operation counts for squared distance functions

A.1. Squared distance functions for cubic Bézier curves

Given a cubic planar Bézier curve $C(t) = (x(t), y(t))$, $0 \leqslant t \leqslant 1$, with four control points $\mathbf{b}_i = (x_i, y_i)$, for $i = 0, 1, 2, 3$, the x-coordinate function is given as $x(t) = (1-t)^3[x_0] + 3(1-t)^2 t[x_1] + 3(1-t)t^2[x_2] + t^3[x_3]$, and its squared function is a Bézier function of degree 6:

$$\begin{aligned} x(t)^2 = {} & (1-t)^6\big[x_0^2\big] + 6(1-t)^5 t[x_0x_1] + 15(1-t)^4t^2\big[0.4x_0x_2 + 0.6x_1^2\big] + 20(1-t)^3t^3[0.1x_0x_3 + 0.9x_1x_2] \\ & + 15(1-t)^2t^4\big[0.4x_1x_3 + 0.6x_2^2\big] + 6(1-t)t^5[x_2x_3] + t^6\big[x_3^2\big]. \end{aligned}$$

Note that the seven control coefficients can be computed using 16 multiplications and 3 additions. Similarly, the squared distance function for the curve is given as follows:

$$\begin{aligned} \big\|C(t)\big\|^2 = {} & x(t)^2 + y(t)^2 \\ = {} & (1-t)^6\big[x_0^2 + y_0^2\big] + 6(1-t)^5t[x_0x_1 + y_0y_1] + 15(1-t)^4t^2\big[0.4(x_0x_2 + y_0y_2) + 0.6\big(x_1^2 + y_1^2\big)\big] \\ & + 20(1-t)^3t^3\big[0.1(x_0x_3 + y_0y_3) + 0.9(x_1x_2 + y_1y_2)\big] + 15(1-t)^2t^4\big[0.4(x_1x_3 + y_1y_3) \\ & + 0.6\big(x_2^2 + y_2^2\big)\big] + 6(1-t)t^5[x_2x_3 + y_2y_3] + t^6\big[x_3^2 + y_3^2\big], \end{aligned}$$

which can be constructed as a Bézier polynomial function $D^2(t)$ using 26 multiplications and 13 additions.

Now the squared distance function for a cubic space Bézier curve can be computed using 36 multiplications and 23 additions. For a cubic rational planar Bézier curve $C(t) = (X(t), Y(t), W(t))$, represented in a homogeneous coordinate, its squared distance function $D^2(t) = (X(t)^2 + Y(t)^2)/W(t)^2$ is a rational Bézier function of degree 6, which can be constructed using 47 multiplications, 16 additions, and 7 divisions. (The additional 7 divisions are needed to get the control coefficients of $D^2(t)$ by dividing each control coefficient of $X(t)^2 + Y(t)^2$ by the corresponding coefficient of $W(t)^2$.) Similarly, for a cubic rational space Bézier curve, it requires 57 multiplications, 26 additions, and 7 divisions.

A.2. Differential of squared distance function

The local extremes for the squared distance function $D^2(t) = \|C(t)\|^2$ can be computed by solving the constraint equation: $\langle C(t), C'(t)\rangle = 0$. At first, a direct multiplication of $C(t)$ and $C'(t)$ may look a reasonable approach to constructing the Bézier representation of $\langle C(t), C'(t)\rangle$. However, because of the asymmetry of $C(t)$ and $C'(t)$, it is not the case.

For a cubic Bézier curve $C(t)$ with its squared distance function $D^2(t)$ with 7 control coefficient f_i, $i = 0, \ldots, 6$, the function $\frac{1}{3}\langle C(t), C'(t)\rangle$ can be computed as a Bézier polynomial of degree 5 with 6 control coefficients $f_{i+1} - f_i$, for $i = 0, \ldots, 5$.

References

Barton, M., Hanniel, I., Elber, G., Kim, M.-S., 2010. Precise Hausdorff distance computation between polygonal meshes. Computer Aided Geometric Design 27 (8), 580–591.

Barton, M., Elber, G., 2011. Spiral fat arcs – Bounding regions with cubic convergence. Graphical Models 73 (2), 50–57.

Chen, X.-D., Yong, J.-H., Wang, G., Paul, J.-C., Xu, G., 2008. Computing minimum distance between a point and a NURBS curve. Computer-Aided Design 40 (10–11), 1051–1054.

Chen, X.-D., Chen, L., Wang, Y., Xu, G., Yong, J.-H., 2009. Computing the minimum distance between Bézier curves. Journal of Computational and Applied Mathematics 230 (1), 294–310.

do Carmo, M., 1976. Differential Geometry of Curves and Surfaces. Prentice-Hall, Inc., NJ, USA.

Elber, G., Kim, M.-S., 2001. Geometric constraint solver using multivariate rational spline functions. In: Proc. of the Sixth ACM Symposium on Solid Modeling and Applications, pp. 1–10.

Hu, S.-M., Wallner, J., 2005. A second order algorithm for orthogonal projection onto curves and surfaces. Computer Aided Geometric Design 22 (3), 251–260.

Gilbert, E., Johnson, D., Keerthi, S., 1988. A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Transactions on Robotics and Automation 4 (2), 193–203.

IRIT, 2009. IRIT 10.0 User's Manual, Technion. http://www.cs.technion.ac.il/~irit.

Johnson, D., 2005. Minimum distance queries for haptic rendering, PhD thesis, Computer Science Department, University of Utah, 2005.

Kim, Y.-J., Oh, Y.-T., Yoon, S.-H., Kim, M.-S., Elber, G., 2010. Precise Hausdorff distance computation for planar freeform curves using biarcs and depth buffer. The Visual Computer 26 (6–8), 1007–1016.

Klosowski, J., Held, M., Mitchell, J., Sowizral, H., Zikan, K., 1998. Efficient collision detection using bounding volume hierarchies of k-DOPs. IEEE Transactions on Visualization and Computer Graphics 4 (1), 21–37.

Lin, M.C., Canny, J., 1991. A fast algorithm for incremental distance calculation. In: IEEE Int. Conf. Robot. Automat., Sacramento, CA, pp. 1008–1014.

Lin, M.C., Gottschalk, S., 1998. Collision detection between geometric models: A survey. In: Proc. of IMA Conference on Mathematics of Surfaces, pp. 37–56.

Lin, M.C., Manocha, D., 2004. Collision and proximity queries. In: Goodman, J.E., O'Rourke, J. (Eds.), Handbook of Discrete and Computational Geometry. 2nd ed. Chapman & Hall/CRC, pp. 787–807.

Liu, X.-M., Yang, L., Yong, J.-H., Gu, H.-J., Sun, J.-G., 2009. A torus patch approximation approach for point projection on surfaces. Computer Aided Geometric Design 26 (5), 593–598.

Ma, Y.L., Hewitt, W., 2003. Point inversion and projection for NURBS curve and surface: control polygon approach. Computer Aided Geometric Design 20 (2), 79–99.

Nishita, T., Sederberg, T.W., Kakimoto, M., 1990. Ray tracing trimmed rational surface patches. Computer Graphics 24 (4), 337–345.

Press, W., Teukolsky, S., Vetterling, W., Flannery, B., 2007. Numerical Recipes: The Art of Scientific Computing, third edition. Cambridge University Press.

Peternell, M., 2000. Geometric properties of bisector surfaces. Graphical Models 62 (3), 202–236.

Sederberg, T.W., Nishita, T., 1990. Curve intersection using Bézier clipping. Computer-Aided Design 22 (9), 337–345.

Selimovic, I., 2006. Improved algorithms for the projection of points on NURBS curves and surfaces. Computer Aided Geometric Design 23 (5), 439–445.

Tang, M., Lee, M., Kim, Y.J., 2009. Interactive Hausdorff distance computation for general polygonal models. ACM Transactions on Graphics (SIGGRAPH '09) 28 (3).