Tracking Point-Curve Critical Distances*

Xianming Chen, Elaine Cohen, and Richard F. Riesenfeld

School of Computing, University of Utah, Salt Lake City, UT 84112, USA,

Abstract. This paper presents a novel approach to continuously and robustly tracking critical (geometrically, perpendicular and/or extremal) distances from a moving plane point $p \in \mathbb{R}^2$ to a static parametrized piecewise rational curve $\gamma(s)$ ($s \in \mathbb{R}$). The approach is a combination of local marching, and the detection and computation of global topological change, both based on the differential properties of a constructed implicit surface; *it does not use any global search strategy* except the initialization.

Implementing the mathematical idea from singularity community, we encode a particular critical distance as a point $p_s = (p, s)$ in the so-called augmented parametric space $\mathbb{R}^3 = \mathbb{R}^2 \times \mathbb{R}$, and the totality of point p_s's (when p moves over the whole plane $\mathbb{R}^2$) as an implicit surface $\mathcal{I}$ in $\mathbb{R}^3$. In most situations, when p is perturbed in the plane, all of its corresponding critical distances, are only evolved, without structural change, by marching on a sectional curve on $\mathcal{I}$. However, occasionally, when the perturbation crosses the evolute of γ, there will be a transition event when a pair of p's current critical distances is annihilated, or a new pair is created and added to the set of p's critical distances. To *safely* eliminate any global search for critical distances, we develop *robust* and efficient algorithm to perform the detection and computation of transition events.

Extra transition events due to various curve discontinuities are also investigated. Our implementation assumes B-spline representation for the curve and has interactive speed even on a lower end laptop computer.

1 Introduction

Given a plane point p and a closed plane curve $\gamma(s) : \mathbb{R} \to \mathbb{R}^2$, the **squared distance function** $f : \mathbb{R} \to \mathbb{R}$ is defined as,

$$f = (\gamma - p)^2. \tag{1}$$

Following the convention of [5], we call a distance from the plane point $\boldsymbol{p}$ to the curve **foot point** $\boldsymbol{\gamma}(s)$ an $\boldsymbol{\mathcal{A}_n}$ **distance**, if $f^{(n+1)}(s)$ is the first non-vanishing derivatives at s. An $\mathcal{A}_n$ distance is called a **critical distance** if $n > 0$, and it is further classified as **regular** if $n = 1$, or **degenerate with multiplicity** $\boldsymbol{n}$

* This work was supported in part by ARO (DAAD19-01-1-0013), NIH (571495), NSF (EIA0121533), and NSF (CCR0310705). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

if $n > 1$. We will generalize critical distances, later in Section 8, to include local extremal distances from p to C^0 break points on a piece-wise smooth curve. In this paper, we encode a critical distance from p to the curve point $\gamma(s)$ as (p, s), and consequently regard it as a point in $\mathbb{R}^3 = \mathbb{R}^2 \times \mathbb{R}$ (cf. Fig. 1b).

Critical distances, especially when generalized to those between a space point and a surface, are of primary interest to many geometric computation including minimal distance computation (e.g., [11, 12]), collision detection (e.g. [18, 19]), and with applications in motion planning and haptic rendering. It also has close relation to medial axis transformation [2] and Voronoi diagrams (e.g., see [6] for the closed smooth curve case). In this paper, we consider two issues, namely, *computing all the topological changes to the critical distances*, and *evolving critical distances if there is no topological change*. By *topology of critical distances*, we mean the total number of critical distances, and the type of each one (i.e., whether it is local minimal, local maximal, or degenerate with certain multiplicity). Notice that, to track the critical distances, it has to start with some given initial plane point position and *all* the corresponding critical distances. This is typically done by solving Eq. (2) (see Section 3) using some constraint solver as discussed in [26, 8]. We will not go into more details on this, and simply assume that the global initialization, with *no missing* critical distances, is given.

In this paper, we are especially interested in the *topological change* of critical distances, called a **transition event**. Mathematically, this is related to singularity and catastrophe theories [1, 23, 14, 22]. In the computer community, most recently, [5] defines the extended curve evolute to serve as the transition set of critical distances on *piecewise smooth* curves, especially deriving *algebraically* all the unfolding formulas for degenerate critical distances. This paper deals with practical implementation issues, including especially the robust and efficient detection of transition events. The totality of critical distances, for all moving points, is formulated as an implicit surface $\mathcal{I}$ in the augmented parametric space $\mathbb{R}^3 = \mathbb{R}^2 \times \mathbb{R}$, and subsequently, the evolution and transition of critical distances are performed using first and second order shape computation on $\mathcal{I}$, respectively.

2 Motivation

We will present a set of algorithms to exactly, continuously, and robustly track point-curve critical distances. For our implementation, the user is allowed to move the point (by mouse) on the plane without any restriction, especially no predefined trajectory, and the critical distances from the point to the static curve are updated interactively. The tracking of critical distance is exact in the sense that there is no polygonization of the curve as is the dominating traditional approach for distance and collision queries. While point-curve distance is important on its own, we are mostly motivated by its future extension to the surface case, that is, tracking the point-surface or surface-surface critical distances, and our ultimate goal is the continuous distance tracking between two trimmed NURBS models both under either rigid motion or more general deformation. The current state-of-the-art in either haptics rendering or motion

planning is mainly of discrete and approximate nature [9, 20, 13]. [10, 21] did work directly on continuous NURBS models, however, just like the situation of polygonal models, global minimal distance search still has to be conducted concurrently (or periodically for the consideration of computation cost, and with the risk of possible wrong haptic rendition).

The transition event of critical distances is important for two reasons namely, *robustness* and *efficiency*. If every transition event is detected, and the corresponding topological change of critical distances is computed subsequently, we are guaranteed of *not missing* any critical distances, and assured of *robustly* reporting the global minimum or maximum. Furthermore, this also eliminates typically expensive (compared to local updating) global search, which gives us efficiency benefit.

For the point-curve case, the transition set is identified either with the evolute of the curve if the considered curve is piecewise smooth with at least C^2 continuity at its break points [14], or with the extended evolute if the piecewise smooth curve has its break points of at least C^0 continuity [5]. For the point-surface case, the transition set is the two focal surfaces [15, 22]. More complex situations arise for the point-model case, when the model consists of multiple trimmed NURBS surfaces with C^0 continuity between surfaces, and for the even more difficult surface-surface and model-model cases. On the other hand, we believe that the basic idea of replacing the global extremal distance search with a robust transition detection and computation, has a straightforward extension to higher dimension, and we present the current work as a first step toward that more ambitious goal.

The rest of the paper is organized as follows. Section 3 presents the problem formulation in the augmented parametric space $\mathbb{R}^3$. Section 4 performs the evolution of the critical distance by marching locally on a sectional curve $\mathcal{I}_\delta$ on $\mathcal{I}$. An osculating circle based correction algorithm is presented in Section 4.1. Section 5 computes the newly created pair of critical distances by contouring the local osculating parabola to $\mathcal{I}_\delta$. Exploiting the fact that the evolute has rational B-spline representation, the robust and efficient *detection* of transition events is investigated in Section 6 using bounding volume tree of the curve evolute. Section 7 presents a simple way to classify the transition event by looking at the sign of $\kappa\kappa'$. To apply our approach to realistic curve models, Section 8 develops algorithms for extra transition events due to various curve discontinuities. After examples in Section 9, conclusions are presented in Section 10.

3 Implicit Surface Formulation in the Augmented Parametric Space

The condition for critical distances between point $p \in \mathbb{R}^2$ and plane curve $\gamma(s) \in \mathbb{R}^2$ is,

$$f' = (\gamma(s) - p) \cdot \gamma(s)' = 0. \tag{2}$$

Regarding the LHS of Eq. (2) as a function of $g = f' : \mathbb{R}^3 \to \mathbb{R}$, the locus of all critical distances, as points (p, s) in $\mathbb{R}^3$, is the zero set of g. The Jacobian of g,

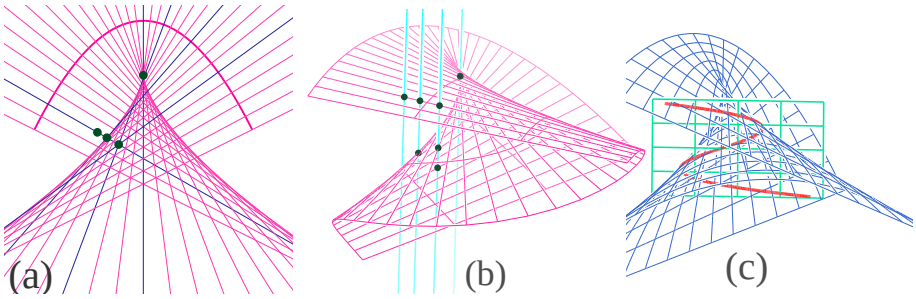

Fig. 1. Implicit Surface Formulation of Point-Curve Critical Distances
(a) shows the normal bundle to a parabola curve; also shown are 4 plane points with their corresponding critical distances (shown as perpendicular lines to the curve). From left to right, they have one regular critical distance (CD), one regular plus another degenerate (with multiplicity 2) CD's, 3 regular CD's, and a degenerate (with multiplicity 3) CD to the curve, respectively. Lifting into the 3-space, in (b), the corresponding vertical lines pierce $\mathcal{I}$ once, pierce once and touch once (on the fold), pierce three times, and pierce once (at the cusp of the fold), respectively. Finally a sectional curve of $\mathcal{I}$ is shown in (c).

$$\mathcal{J} = (\nabla g) = \begin{pmatrix} -\gamma_0' \\ -\gamma_1' \\ (\gamma - p) \cdot \gamma'' + \|\gamma'\|^2 \end{pmatrix}, \tag{3}$$

always has full rank (i.e., 1) under the assumption that γ is regular; thus, especially with a regular value of 0, the zero set of g is a 2-manifold in $\mathbb{R}^3$, denoted as $\mathcal{I}$ hereafter. Geometrically, $\mathcal{I}$ is the lifted normal bundle to the curve (Fig. 1 (a)&(b)), and is called the catastrophe surface in [14].

Notice that the normal to $\mathcal{I}$ actually has the same expression as the RHS of Eq. (3), and can be succinctly written as

$$N_{\mathcal{I}} = -\gamma' + \mathfrak{D} e_s. \tag{4}$$

where e_s is the unit vector along the vertical s-axis in $\mathbb{R}^3$, $\gamma' \in \mathbb{R}^2$ is regarded as $\gamma' \in \mathbb{R}^3$ in a natural way (i.e., the last component is 0), and $\mathfrak{D} = (\gamma - p) \cdot \gamma'' + \|\gamma'\|^2$.

Finally, we recall a few identities which are used later in this paper.

$$e_s \times a = a_{\mathfrak{r}}, \qquad a \times b = (a_{\mathfrak{r}} \cdot b) e_s, \qquad (a_{\mathfrak{r}})_{\mathfrak{r}} = -a, \qquad a_{\mathfrak{r}} \cdot b_{\mathfrak{r}} = a \cdot b \tag{5}$$

where a and b are vectors in $\mathbb{R}^2$, and the subscript $\mathfrak{r}$ denotes 90 degree rotation around positive s-axis.

4 Evolution

Given any perturbation of p, δ_p , we want to evolve all critical distances corresponding to p to those corresponding $\hat{p} = p + \delta_p$. Geometrically, critical distance (p, s)'s, for a fixed p, are the intersection points of $\mathcal{I}$ with the vertical line passing through p, while the critical distance $(p + \delta_p, s + \delta_s)$'s are the intersection points of $\mathcal{I}$ with the vertical line passing through $\hat{p} = p + \delta_p$ (cf. Fig. 1b). In the following, we only consider *locally* a particular critical distance (p, s).

First, construct the sectional curve, $\mathcal{I}_\delta$, on $\mathcal{I}$, that is, the intersection of $\mathcal{I}$ with a vertical plane $\mathcal{P}$ passing through both plane points p and $p + \delta_p$ (Fig. 1c).

By Eq. (4), and since $\mathcal{P}$ has normal $N_{\mathcal{P}} = (\delta_p)_{\mathfrak{r}}$, the tangent to $\mathcal{I}_\delta$ is,

$$T = N_{\mathcal{P}} \times N_{\mathcal{I}} = (\delta_p)_{\mathfrak{r}} \times (-\gamma' + \mathfrak{D} e_s) = -(\delta_p)_{\mathfrak{r}} \times \gamma' + \mathfrak{D}(\delta_p)_{\mathfrak{r}} \times e_s.$$

By Eq. (5),

$$T = (\delta_p \cdot \gamma')\ e_s + \mathfrak{D}\ \delta_p. \tag{6}$$

Therefore, the critical distance (p, s) is evolved to $(p + \delta_p, s + \frac{\delta_p \cdot \gamma'}{\mathfrak{D}})$, assuming that $\mathfrak{D}$ does not vanish (otherwise, there has to be a transition event, which is discussed later in Section 5).

4.1 Correction

The local evolution of critical distances, as just described, essentially uses tangent plane approximation to the implicit surface $\mathcal{I}$, and there will be accumulated error over time. We develop a curvature-based correction algorithm in this section.

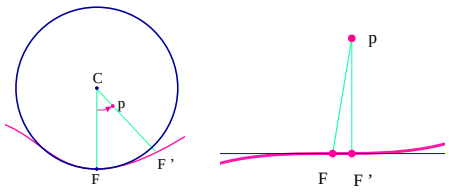

Fig. 2. Circle/Tangent Approx.

The basic idea is to approximate the local curve with its osculating circle (Fig. 2(a)). Suppose, for a plane point p, one of its approximate critical distances, under consideration, has a foot point F. However, p is not really on the normal line to F; instead, it deviates from the normal line by an angle $d\alpha = \angle FCp$, where C is the curvature center corresponding to F. Recall that, for a plane curve, the signed curvature is the rate of change α with respect to arc length, i.e., $\kappa = \frac{1}{\|\gamma'\|}\frac{d\alpha}{ds}$; so,

$$ds = \frac{d\alpha}{\kappa \|\gamma'\|}. \tag{7}$$

If F is near a curvature zero point, we replace the osculating circle based correction algorithm by a tangent based one (Fig. 2(b)),

$$ds = \frac{\|F' - F\|}{|\gamma'\|} = \frac{(p - F) \cdot \gamma'}{\|\gamma'\|^2}. \tag{8}$$

5 Transition

If $\mathfrak{D} = 0$, $\mathcal{I}_\delta$ is locally vertical at the considered $(p, s) \in \mathbb{R}^3$ (cf. Eq. (6)), then there is no way of evolving (p, s) to the next approximate critical distance by following $\mathcal{I}_\delta$ tangentially. Mathematically, the locus of such points (p, s) forms the fold singularity of projection of $\mathcal{I}$ on s-axis. For C^2 curve case, the projection of the fold is actually the evolute of γ [14].

When the plane point moves across a point on the evolute, there will be a transition event, i.e. some structural change of the critical distances. Therefore, all points on the evolute (or the extended evolute, for piecewise smooth curves later in Section 8) are called **transition points**. At a transition point, the corresponding critical distance, is degenerate with multiplicity 2. Away from the transition point, the critical distance disappears completely in one direction, and unfolds into *a pair* of critical distances in the other direction. They are called an **annihilation event** and a **creation event**, respectively. Notice that the created pair of critical distances have to be of opposite types (one minimum, and the other maximum). [5] gives detailed algebraic derivations related to transition events.

In this Section, we initialize the created pair of critical distances by second order differential computation on $\mathcal{I}_\delta$, i.e, more specifically, by contouring the local osculating parabola to $\mathcal{I}_\delta$.

Eq. (6) gives the tangent vector field on the curve $\mathcal{I}_\delta$, and it allows us to compute the covariant derivative with respect to itself. At a singular point where $\mathfrak{D} = 0$ and $T = (\delta_p \cdot \gamma')\ e_s + \mathfrak{D}\ \delta_p = (\delta_p \cdot \gamma')\ e_s$ (cf. Eq. (6)), the curvature of $\mathcal{I}_\delta$ is (we denote, generally, $\boldsymbol{\kappa}$ as the unit normal of the curve under consideration, scaled by the signed curvature κ),

$$\begin{aligned}\boldsymbol{\kappa}_{\mathcal{I}_\delta} &= \frac{(T \times \nabla_T T) \times T}{T^4} = \frac{\Big((\delta_p \cdot \gamma')\ e_s \times (\delta_p \cdot \gamma')\frac{\partial T}{\partial s}\Big) \times (\delta_p \cdot \gamma')\ e_s}{(\delta_p \cdot \gamma')^4} \\ &= \frac{\Big(e_s \times \frac{\partial\big((\delta_p \cdot \gamma')\ e_s + \mathfrak{D}\ \delta_p\big)}{\partial s}\Big) \times e_s}{\delta_p \cdot \gamma'} = \frac{\partial \mathfrak{D}}{\partial s}\ \frac{(e_s \times \delta_p) \times e_s}{\delta_p \cdot \gamma'} = \frac{\mathfrak{D}'}{\delta_p \cdot \gamma'}\ \delta_p \qquad (9)\end{aligned}$$

where (see detail in [3]),

$$\mathfrak{D}' = \frac{\partial \mathfrak{D}}{\partial s} = \Big((\gamma - p) \cdot \gamma'' + \|\gamma'\|^2\Big)' = -\|\gamma'\|^2 \frac{\kappa'}{\kappa}. \qquad (10)$$

Assume that p is originally *at a transition point* (i.e., on the evolute), with a degenerate critical distance (p, s) of multiplicity 2, and assume further that p is perturbed by δ_p. If $\frac{\mathfrak{D}'}{(\delta_p \cdot \gamma')} > 0$, then $\boldsymbol{\kappa}_{\mathcal{I}_\delta}$ has the same direction as δ_p by Eq. (9), or the perturbation direction is toward the curved side of $\mathcal{I}_\delta$; therefore, there will be a pair of new critical distances to be created upon this perturbation (cf. Fig 1c). Approximating the local curve of $\mathcal{I}_\delta$ by its osculating parabola,

$$\delta_p = \frac{1}{2}\boldsymbol{\kappa}_{\mathcal{I}_\delta}\delta_s^2 = \frac{1}{2}\frac{\mathfrak{D}'}{(\delta_p \cdot \gamma')}\delta_s^2\ \delta_p,$$

so

$$\frac{1}{2}\frac{\mathfrak{D}'}{(\delta_p \cdot \gamma')}\delta_s^2 = 1.$$

Therefore, the pair of critical distances are $(p + \delta_p, s + \delta_s)$ and $(p + \delta_p, s - \delta_s)$, where δ_s is,

$$\delta_s = \sqrt{\frac{2(\delta_p \cdot \gamma')}{\mathfrak{D}'}}. \tag{11}$$

On the other hand, if the perturbation is away from the curved side of $\mathcal{I}_\delta$, the original critical distance will disappear. The pair of critical distances to disappear or annihilate is evident, given the fact that it is the unique neighboring critical distances (p, s_0) and (p, s_1) with $s \in (s_0, s_1)$.

Notice that, other than $\mathcal{A}_2$ (multiplicity-2) critical distances, there are also $\mathcal{A}_3$ or even higher degenerate ones([5]). $\mathcal{A}_3$ critical distances correspond to *isolated* ordinary cusps on the evolute. and therefore, due to numerical error and/or intentional numerical perturbation, the creation events can be safely assumed to be of only $\mathcal{A}_2$ type. However, for robust tracking algorithm, special implementation is required for the situation when the plane point is very close to a cusp point (other type of cusps may also arise on an extended evolute, cf. Fig. 4). See [3] for detail.

6 Detection of Transition Events

The detection of a transition event is, in principle, not more complicated than a special curve-curve intersection problem [17, 16, 25, 24]. We use the interval subdivision method [16] to do the subdivision, and, in addition, to construct a bounding volume tree (BVT) [1] from the axis aligned bounding boxes resulting from the interval subdivision. For most situations, the BVT allows the intersection algorithm to stop at a very early stage. In this section, the word "intersection" will mean *line-diagonal* intersection, while the word "hit" will refer to the intersection of the line with the box *edges*.

Construct Evolute BVT by Interval Subdivision Interval subdivision requires a pre-processing of breaking the initial curve at any point where a component of the curve derivative vanishes. However, the evolute is not a regular curve, and the pre-precessing also needs to split the evolute at both its asymptote and cusp points (see Fig. 4). See [3] for more implementation details.

Line Hits Axis Aligned Box The first stage of transition detection checks if a parametrized line $\mathcal{L}(t)$, with $\mathcal{L}(0) = p$ and $\mathcal{L}(1) = \hat{p}$, hits an axis aligned box. While this is essentially the typical ray tracing algorithm [27], more hit

[1] Notice that the volume here is the 2-dimensional area on the plane.

information is required for the next stage algorithm; specifically, if there is a hit, the algorithm should compute the following three pieces of information for both near and far hit points.

1. the hit edge.
2. the ratio of hit point with respect to $p\hat{p}$, i.e., parameter t of hit point.
3. the ratio of hit point with respect to the hit edge.

If both hit points are outside of $p\hat{p}$, the line *segment* $p\hat{p}$ can not intersect the box diagonal in any case and therefore there will be no transition event, and consequently, the algorithm stops.

Line Segment Intersects Box Diagonal A leaf node axis aligned box of the BVT, is constructed simply from the two end points of the control polygon resulting from interval subdivision. Throughout this section, by "diagonal", we mean only the diagonal that connects these two end points. Notice that the diagonal segment is supposed to approximate the local curve, and has parameters for both of its ends that are *on* the evolute.

The second stage of the transition detection checks if the line segment intersects the diagonal of the hit box, and, if so, computes the ratio of the intersected point with respect to the two end points of the diagonal. The intersection point is an approximation to the real intersection point of $p\hat{p}$ with the curve evolute, and its parameter is *interpolated* from those of the two diagonal ends (instead of the simple middle point approximation).

Remark 1 An interpolation based approach gives more accurate result than the middle point approximation, which is highly desirable because, *at a transition point*, a perturbation δ_p of the point p will cause a perturbation $\sqrt{\|\delta_p\|}$ of the foot point of the corresponding critical distance. For details see Lemma 1 in [5].

Three generic situations of line-diagonal intersection are illustrated in Fig. 3. It is either a through-intersection, when the near and far hit points are on the opposite edges, or a corner-intersection, when they are on the neighboring edges. On the other hand, considering the relative orientation of the diagonal segment qq' with respect to the line segment pp', it is either that $qq'[i]$ has the same sign as that of $pp'[i]$ for *both* $i = 0$ and $i = 1$, or the sign relations are opposite for $i = 0$ and $i = 1$ (the boolean array $diag[2]$ in Algorithm 1 keeps this information).

The ratio of intersection point w.r.t. the diagonal is computed, based on this classification, and by constructing the auxiliary similar triangles (shaded in Fig. 3). See details in Algorithm 1.

7 Classification of Transition Types

If Algorithm 1 returns true, then the line segment $p\hat{p}$ (i.e., the perturbation) does intersect the evolute and there is going to be a transition event. The transition event is of creation type, if the perturbation is toward curved side of $\mathcal{I}_\delta$, and of

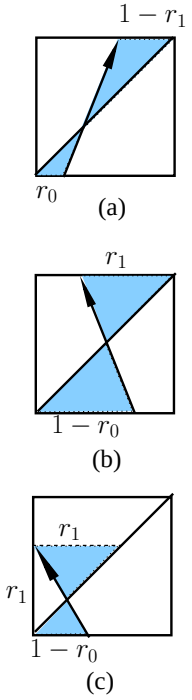

Fig. 3. Generic Line Box-Diagonal Intersection

Input:
diag[**2**] box diagonal direction w.r.t. $\mathbf{p\hat{p}}$
$(\mathbf{e_0}, \mathbf{t_0}, \mathbf{r_0})$ near hit point
$(\mathbf{e_1}, \mathbf{t_1}, \mathbf{r_1})$ far hit point
Output: on *true* return,
t ratio of intersected point w.r.t. $\mathbf{p\hat{p}}$
r ratio of intersected point w.r.t. diagonal
Return:
true if intersected (i.e. $t \in [0, 1]$), *false* otherwise
Begin
If **diag**[**0**] $\neq$ **diag**[**1**]
$\mathbf{t} \Leftarrow (\mathbf{1} - \mathbf{r_0})/(\mathbf{1} - \mathbf{r_0} + \mathbf{r_1})$
$\mathbf{i} \Leftarrow$ *is_horizontal*$(\mathbf{e_0})$? **0** : **1**
diag[**i**] $\Leftarrow$! **diag**[**i**]
Else
If corner cut situation **Return** *false*
$\mathbf{t} \Leftarrow \mathbf{r_0}/(\mathbf{r_0} + \mathbf{1} - \mathbf{r_1})$
$\mathbf{r} \Leftarrow \mathbf{t} \Leftarrow (\mathbf{1}-\mathbf{r})\mathbf{t_0}+\mathbf{r}\mathbf{t_1}$
If $\mathbf{t} \notin [\mathbf{0}, \mathbf{1}]$ **Return** *false*
If case (c) in Fig. 3 $\mathbf{r} \Leftarrow \mathbf{r} * \mathbf{r_1}$
If(! **diag**[**0**]) $\mathbf{r} \Leftarrow \mathbf{1} - \mathbf{r}$
Return *true*
End

Algorithm 1. Line Segment Intersects Box Diagonal

annihilation type otherwise (cf. Fig. 1c). By Eq. (9), the perturbation direction is toward the curved side of the sectional curve $\mathcal{I}_\delta$, or $\boldsymbol{\kappa}_{\mathcal{I}_\delta}$ has the same direction as δ_p, if and only if $\mathfrak{D}'(\gamma' \cdot \delta_p) > 0$; hence, evaluation of $\mathfrak{D}'$ and $\gamma' \cdot \delta_p$ at the transition point will determine the exact transition type. However, by Eq. (10), $\mathfrak{D}'$ only changes sign at isolated curve points where $\kappa\kappa'$ changes sign; therefore, the evaluation of $\mathfrak{D}'$, which involves the third order derivative, is not necessary at run time, provided that all the sign flipping points of $\kappa\kappa'$ are already pre-computed.

The following simple algorithm determines the sign of $\kappa\kappa'$, without even evaluating κ' at any point.

Algorithm 2 Pre-computing signs of $\kappa\kappa'$

1. *Split the curve at all its zero curvature points, critical curvature points, and all break points with continuity* $C^{(<3)}$,
2. *Evaluate only* curvature *at all split points. The evaluation may be not necessary, if the point is zero curvature point; and it may have to be performed twice for left and right limit evaluations, for a* $C^{(<2)}$ *break point.*

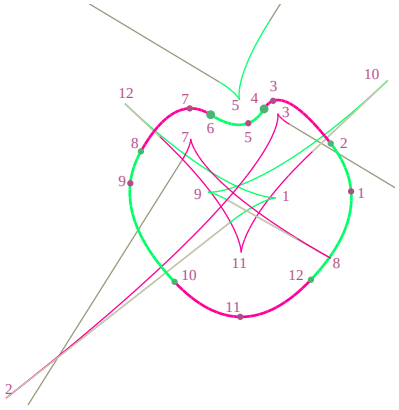

Fig. 4. Flipping Points of $\kappa\kappa'$ of a quadratic B-spline curve

$\boldsymbol{\kappa' = 0}$: $1, 3, 5, 7, 9, 11$
vertices on the curve, ordinary cusps on the extended evolute.

$\boldsymbol{\kappa'_l \kappa'_r < 0}$ **and** $\boldsymbol{\kappa_- \kappa_+ > 0}$: $2, 8, 10, 12$
C^1 breaks on the curve, cusps on the extended evolute.

$\boldsymbol{\kappa'_l \kappa'_r > 0}$ **and** $\boldsymbol{\kappa_- \kappa_+ < 0}$: $4, 6$ C^1 breaks on the curve, asymptotes on the extended evolute.

3. *For each segment, tag the sign of κ the same as that of any of its non zero curvature end point, and the sign of κ' the same as the difference of κ at its two end points.*

The algorithm requires as input the zero and critical curvature points, which are computed using NURBS symbolic computation [7], with degree reduction strategy detailed out in [4]. Fig. 4 shows all the flipping points of $\kappa\kappa'$ for a quadratic B-spline curve.

8 Extra Transition Events at Curve Break Points

In this section, we make extension to our algorithms so that critical distances can be tracked across curve break points of at least C^0 continuity. Notice that the corresponding $C^{(-1)}$ situation is not any more difficult, and is omitted here under the consideration that any curve, as a boundary to some 2D shape, has to be closed.

***C^2* Break Points** First, observe that, by Eq. (6), Eq. (7) and Eq. (11), as long as the curve is C^3, the algorithms are valid. The requirement, however, can be relaxed to C^2. A C^2 point of the curve corresponds to a C^1 point on the implicit surface $\mathcal{I}$, and so does not affect evolution algorithm. On the other hand, it does affect the transition computation based on $\boldsymbol{\kappa}_{\mathcal{I}_\delta}$, because $\boldsymbol{\kappa}_{\mathcal{I}_\delta}$ is C^{-1} by Eq. (9). A simple solution is to evaluate the left and right limits of Eq. (11). However, observing that there are only isolated transition points on the evolute corresponding to curve C^2 break points, this could rarely happen due to numerical error.

***C^1* Break Points** Usually only one point of any lifted normal line is on the fold of $\mathcal{I}$, and the projection of that point to $\mathbb{R}^2$ is on the evolute, or it is the curvature center (cf. Fig. 1b). However, if the considered normal line corresponds

to a C^1 curve point, there will be a whole *segment* of it being on the fold of $\mathcal{I}$, and there will be a creation or an annihilation of a pair of new critical distances when the perturbation crosses any point of that segment. The normal line *segment*, serving as extra transition points, is either the line segment connecting the two (left limit and right limit) curvature centers, or the compliment of it with respect to the whole normal line (see [5] for more details). The important thing to note, though, is that the apparent *transition* event is actually because of two *evolution* events, one performed on the left segment and the other performed on the right segment. Therefore, the following straightforward algorithm will compute the (apparent) transition event or evolution event at a point (p, s) where γ is C^1 at s, given a perturbation of δ_p. In the rest of the paper, a subscript of l (r) denotes the left (right) limit evaluation.

Algorithm 3 Transition/Evolution at C^1 Break

1. *If* $(\delta_s)_l = \frac{\delta_p \cdot \gamma'}{\mathfrak{D}_l} < 0$, $(p + \delta_p, s + (\delta_s)_l)$ *is a perturbed critical distance.*
2. *If* $(\delta_s)_r = \frac{\delta_p \cdot \gamma'}{\mathfrak{D}_r} > 0$, $(p + \delta_p, s + (\delta_s)_r)$ *is a perturbed critical distance.*

A creation/evolution/annihilation event happens if two/one/none perturbed distances are returned from the algorithm.

C^0 Break Points At a C^0 curve break point, there are two normal lines, and each of the lifted ones has some segment on the fold of the implicit surface $\mathcal{I}$. We could have done transition computation directly for C^0 break points. However, as suggested by [5], a C^0 break point can be converted into two (collapsed) C^1 break points by inserting an arc with 0-radius in between the two unfolded break points. The arc has tangent continuity at both its ends, and has positive (negative) infinite curvature if the two tangents at the left and right ends form a right (left) hand rotation. Notice that this is essentially assigning a whole span of normal lines to a C^0 point, generated by right (left) rotating the left limit normal to the right limit normal; consequently, there will be an extra critical point if the plane point is on any of these normal lines.

9 Examples

The two examples in this section are snapshots taken from dynamically tracking critical distances from a moving (user interactive with a mouse) point to a static curve. Demo videos are accessible by following the link
http://www.cs.utah.edu/∼xchen/papers/more.html

Fig. 5 gives an example of continuously tracking critical distances on a cubic B-spline curve, with 6 snapshots taken from the animation. The plane point is shown in dark square box, and foot points are shown in filled circles, while the corresponding points on the evolute (light gray colored) are shown in non-filled circles. There are 5 transition events occurred at some point between each pair of neighboring snapshots. The first transition annihilates a pair of critical distance,

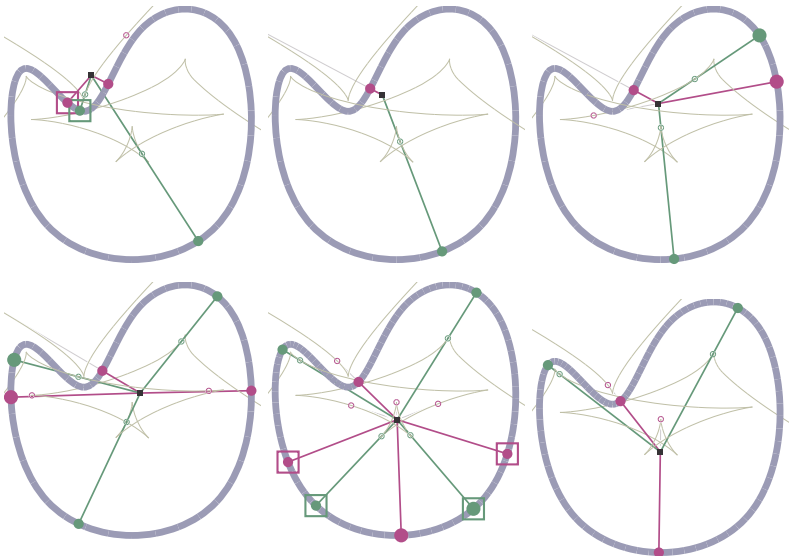

Fig. 5. Example 1: Tracking Critical Distances on a C^2 B-spline Curve

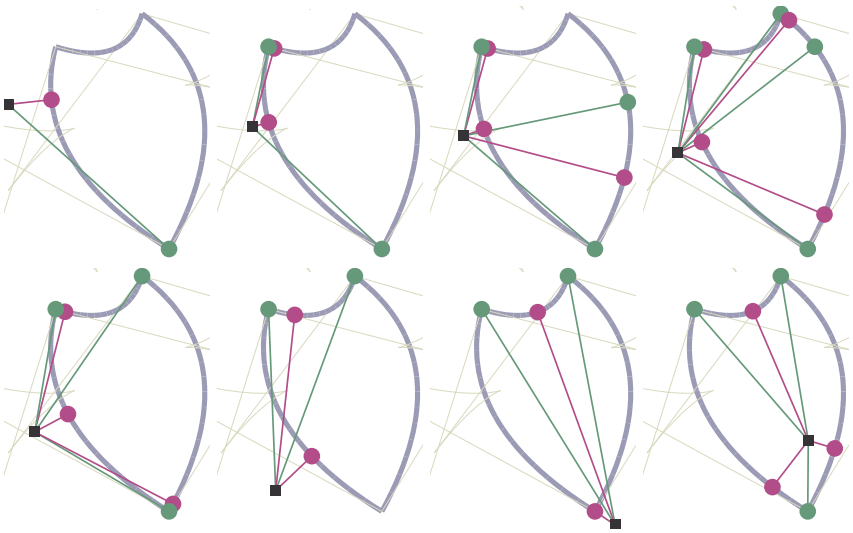

Fig. 6. Example 2: Tracking Critical Distances on a C^0 B-spline Curve

while the last one is actually two transition events, each of which annihilates a pair of critical distances. All the rest of the transitions create a pair of critical distances. The pair to be annihilated is shown in boxes, while the created pair in larger filled circles.

Fig. 6 shows the extremal distance tracking on a C^0 B-spline curve. Only part of the extended evolute curve is shown in very light color (see [5, 3] for details).

10 Conclusion

In this paper, we have formulated the totality of critical distances from a plane point to a curve, as an implicit surface $\mathcal{I}$ in the augmented parametric space. The evolution and the transition of critical distances are achieved by first and second order differential computation on $\mathcal{I}$, respectively. The detection of transition is robustly and efficiently implemented using bounding volume tree of the curve evolute. Extra transition events, corresponding to curve break points, are also computed.

So far we have not mentioned global minimal/maximal distance tracking because, to robustly track *global minimal/maximal* distance, all the *local critical* distances have to be tracked. Because critical distances always comes with alternating types [2](cf.Section 8), the global minimal/maximal distance can be computed at run time by searching and comparing every other current critical distance.

Notice that global *minimal* distance tracking might also be done using medial axis transformation (MAT) on the curve, and probably more efficiently. This is a possible topic for our future work.

With this continuous tracking of point-curve critical distances efficiently achieved using local marching, robust transition detection, and occasional transition computation, *without any traditional global searching*, our future agenda, is to extend the approach to and design robust algorithms for the point-model case and the model-model case.

References

[1] V. Arnold, *Catastrophe Theory*, 3 edition, Springer-Verlag, 1992.

[2] H. Blum, "A transformation for extracting new descriptors of shape," *Models for the perception of speech and visual forms*. 1967, pp. 362–380, MIT Press.

[3] X. Chen, "Dynamic Geometric Computation by Singularity Detection and Shape Analysis," *Ph.D. Thesis Manuscript*, 2006.

[4] X. Chen, R. Riesenfeld, and E. Cohen, "Degree Reduction Strategies for NURBS Symbolic Computation," *Proceedings of IEEE Shape Modeling and Applications 2006: 182-193*, 2006.

[2] a degenerate critical distance of neither minimum nor maximum could never happen in practical implementation due to numerical error or by intentional ϵ-perturbation

[5] X. Chen, R. Riesenfeld, and E. Cohen, "Extended Curve Evolute as the Transition Set of Distance Functions," *Under submission*, 2006.

[6] J. J. Chou, "Voronoi diagrams for planar shapes," *IEEE Computer Graphics and Applications*, vol. 15, no. 2, 1995, pp. 52–59.

[7] G. Elber, "Free Form Surface Analysis using a Hybrid of Symbolic and Numeric Computation," *Ph.D. thesis, University of Utah, Computer Science Department*, 1992.

[8] G. Elber and M.-S. Kim, "Geometric constraint solver using multivariate rational spline functions," *ACM Symposium on Solid Modeling and Applications*, 2001, pp. 1–10.

[9] A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "A Framework for Fast and Accurate Collision Detection for Haptic Interaction," *IEEE VR 1999*, 1999.

[10] T. V. T. II and E. Cohen, "Direct Haptic Rendering Of Complex Trimmed NURBS Models," *ASME Proc. 8th Annual Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Nov. 1999.

[11] D. Johnson and E. Cohen, "A framework for efficient minimum distance computations," *Proc. IEEE Intl. Conf. Robotics and Automation*, May 1998, pp. 3678–3684.

[12] D. Johnson and E. Cohen, "Bound Coherence for Minimum Distance Computations," *IEEE Proc. International Conference on Robotics and Automation*, 1999.

[13] D. E. Johnson, P. Willemsen, and E. Cohen, "A Haptic System for Virtual Prototyping of Polygonal Models," *DETC 2004*, 2004.

[14] J.W.Bruce and P.J.Giblin, *Curves And Singularities*, 2 edition, Cambridge University Press, 1992.

[15] J. J. Koenderink, *Solid Shape*, MIT press, 1990.

[16] P. Koparlar and S. P. Mudur, "A new class of algorithms for the processing of parametric curves," *Computer-Aided Design*, vol. 15, 1983, pp. 41–45.

[17] J. M. Lane and R. F. Riesenfeld, "A theorectical development for the computer generation and display of piecewise polynomial surfaces," *IEEE Trans. PAMI*, vol. 2, 1980, pp. 35–46.

[18] M. C. Lin, "Efficient Collision Detection for Animation and Robotics," *Ph.D. thesis,* University of California, Berkeley, 1993.

[19] M. C. Lin and D. Manocha, "Collision and proximity queries," *Handbook of discrete and computational geometry*, 2004, pp. 787–807.

[20] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling," *SIGGRAPH 1999*, 1999, pp. 401–408.

[21] D. D. Nelson, D. Johnson, and E. Cohen, "Haptic Rendering of Surface-to-Surface Sculpted Model Interaction," *ASME Proc. 8th Annual Symp. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Nov. 1999.

[22] I. R. Porteous, *Geometric Differentiation: For the Intelligence of Curves and Surfaces*, 2 edition, Cambridge University Press, 2001.

[23] P. T. Saunders, *An Introduction to Catastrophe Theory*, 2 edition, Cambridge University Press, 1980.

[24] T. W. Sederberg and T. Nishita, "Curve intersection using Bézier clipping," *Computer-Aided Design*, vol. 22, 1990, pp. 538–549.

[25] T. W. Sederberg and S. R. Parry, "A comparison of curve-curve intersection algorithms," *Computer-Aided Design*, vol. 18, 1986, pp. 58–63.

[26] E. C. Sherbrooke and N. M. Patrikalakis, "Computation of the solutions of nonlinear polynomial systems," *Computer Aided Geometric Design*, vol. 10, no. 5, 1993, pp. 379–405.

[27] P. Shirley and R. K. Morley, *Realistic Ray Tracing*, 2 edition, A K Peters Ltd., 2003.