An Inexpensive Bounding Representation for Offsets of Quadratic Curves

Erik Ruf*
Microsoft Research

Abstract

We describe a simple mechanism for bounding the portion of the plane lying between a quadratic Beizer curve segment and its offset curve at distance d. Instead of comprising one or more partial bounding polygons, our representation consists of only a single approximate offset curve segment, also in quadratic Bezier form. Evaluated on a corpus of real-world curves, this technique avoids 68-99% of antialias-distance queries and 41-96% of brush-parameter queries. A proof of correctness is provided.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations;

Keywords: offset curve, bounding technique, acceleration

1 Introduction

The distance from a point to a planar curve is useful in a number of applications, including computer graphics (antialiasing, brush effects), manufacturing (tool planning), and road navigation. In many cases, computing the distance value (more generally, the closest point on a curve segment) involves executing an expensive iterative refinement algorithm on each sample point within a suitable bounding region. This bounding region must be conservative, in that, for a chosen maximum distance d, all points within d of the curve must lie within the bounding region. Since only points within the bounding region need be examined, solution costs are reduced as the bounding region becomes more precise. At the same time, increased precision typically comes at the cost of a more complex bounding region representation, which may require more storage space and more time to construct and/or evaluate. This paper describes a new point in the design space that yields better precision than the standard rectangular approximation, but requires little additional storage, construction, or evaluation time.

Given that we are interested only in solutions for query points within distance d of the curve, an ideal bounding representation would consist of either the original curve and a precise exterior offset curve at distance $+d$ from the curve (e.g., for antialiasing of a glyph outline) or a pair of precise interior and exterior offset curves at distances $+d$ and $-d$ (e.g., a brush of radius d whose center moves along the curve). A query point would then be relevant if and only if it lies between the two curves. This test is easily performed by evaluating the implicit equations describing the curves and examining the signs of the resulting values. Unfortunately, the precise offset curve for even a low-order polynomial can be difficult to obtain and expensive to evaluate.[1] Our approach stems from two observations:

1. The offset curve isn't an end product; it's only an accelerator for an underlying precise distance computation. Thus, even an imprecise offset curve can be useful, provided that it is conservative and its cost per avoided query is kept sufficiently low and,

2. For small d, we can obtain a useful approximate offset curve from the original control triangle in constant space and time. We accomplish this by extending the curve segment into an axis-symmetric form, then offsetting the endpoints of the resulting control triangle.

This paper explores these observations in the context of a single curve type: the quadratic Bezier curve [Farin 2002]. We begin by motivating the use of curve segments, rather than polygons, as bounding regions. We then describe our algorithm for constructing approximate offset curves, and demonstrate the utility of such curves in two scenarios involving real-world curves. We conclude with discussions of related and future work. In the Appendix, we argue that our approach is sound.

2 Algorithm

2.1 Basics

In this paper, we treat only quadratic Bezier curves, which are specified via a control triangle $p_0p_1p_2$. This triangle describes both an infinite planar curve and a segment of that curve that begins at p_0 and ends at p_2. We can construct an implicit equation $C(p) = 0$ for the infinite curve, as well as a singly-parameterized formula $F(t)$ such that $F(0) = p_0$, $F(1) = p_1$, and $\forall t, C(F(t)) = 0$.

2.2 Bounding regions for acceleration

The simplest polygonal acceleration structure for d-bounded distance queries on a Bezier curve segment is the control triangle itself (extended to include all points within d of any control point). Similarly, we can obtain the optimum axis-aligned rectangular bounding region by finding the x- and y- extremes of the segment, constructing the minimal rectangle that encloses these points, then extending its corner points outward by d.

Various higher-precision techniques exist, and are treated in the Related Work section of this paper on page 4. Here, we consider the tradeoffs inherent in choosing an acceleration structure. The precision of the bounding-polygon approach can be improved by increasing the number of vertices, at the cost of additional storage usage and the increasing number of half-plane tests required to discard a query point as irrelevant.

The usual alternative to a single, complex polygonal bounding region is a collection of small, simple polygonal bounding regions (usually rectangles). By repetitively subdividing the query space, we can obtain ever-smaller bounding polygons. This scheme can achieve an arbitrary level of precision, but at an additional costs, including

1. **space**. Somehow, we must map query points to subregions, and this requires a data structure, usually a grid or tree.

*e-mail: erikruf@microsoft.com

[1] Anton et al. [2005] find that a parabola's offset curve is of degree 6.

HPG 2011, Vancouver, British Columbia, Canada, August 5 – 7, 2011.
© 2011 ACM 978-1-4503-0896-0/11/0008 $10.00

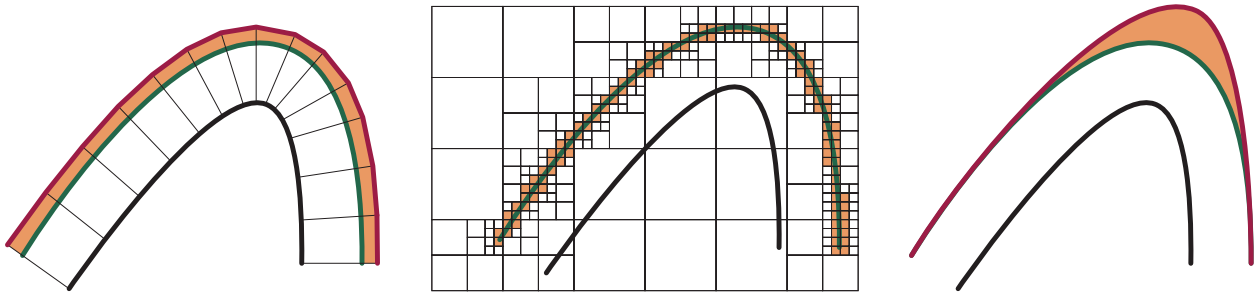

Figure 1: Representations for offset regions. *We show (a) a piecewise-linear normal offset, (b) a quadtree decomposition, and (c) a quadratic offset curve. In all cases, the initial quadratic curve is black and the precise offset curve being approximated is in green. Beige sections lying outside the precise curve represent imprecision in the approximate representation. Improving precision in (a) or (b) is accomplished by adding edges or rectangles, while improving (c) requires improving the fit of the approximate curve by altering its parameters and/or increasing its degree.*

2. **query time**. Each time we perform a query, we must traverse the data structure, performing tests to direct our path toward a leaf note (usually a boolean truth value or a polygon inclusion test). Depending on the data structure used, we may have to perform tests against multiple bounding polygons.

3. **setup time**. For some curve segments , such as those describing glyph contours, it may make sense to statically construct an efficient acceleration data structure, as this cost can be amortized over many queries. However, dynamic applications (e.g., authoring tools) that construct curves "on the fly" may not be able to repay the cost of accelerator data structure construction for short-lived curve segments.

Rather than increasing the complexity or number of polygons in the acceleration structure, we instead seek to improve the fidelity of the *segments* comprising the bounding region. Using one or more curve segments rather than numerous line segments allows greater precision at a small cost in representation space and evaluation time. Figure 1 graphically depicts instances of the polygonal, multi-polygonal, and curve-based approaches.

2.3 Motivation: offset curves as bounding regions

The query points relevant to a particular d-bounded distance query on an infinite curve C can be succinctly described as those lying between the curve and an *approximate offset curve*, all of whose points lie at least d from the original curve C. Provided that both the original and offset curves have implicit representations, testing the relevance of a query reduces to evaluating both implicit formulas on the query point and comparing the signs of the resulting values.

A query on a finite curve segment S adds the additional constraint that only query points whose closest curve point lies on S need be considered. Thus, for efficiency's sake, it is useful (though not necessary) to exclude queries whose results cannot lie on S. This is easily accomplished via any of the simple polygonal tests described above (our implementation uses the d-extended rectangle); other means, such as explicitly checking distances from the query points to the endpoints, or executing line-side tests with respect to the normal vectors at the endpoints, are also possible.

Thus, deciding whether a given query point need be sent to the iterative distance analysis requires, at most, two implicit curve evaluations and a rectangle inclusion test. The rectangle test can sometimes be elided, as any enclosing rectangle (e.g., a glyph bounding box or a tile in a grid decomposition of a graphical object) can be used to limit the infinite region delineated by the offset curves to a small portion of the plane.

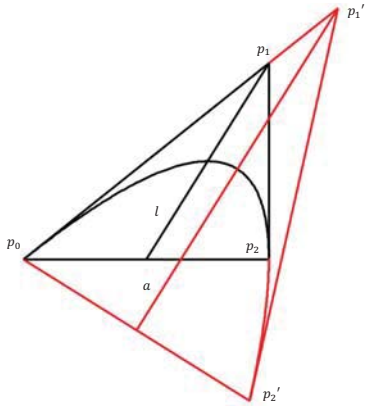

Figure 2: Constructing the axis-aligned control triangle. *The black lines show the initial control triangle, the "midline" connecting p_1 and the center of p_0p_2, and the curve segment. The red line parallel to the midline is the curve axis; reflecting p_0 across it yields p_2'; extending the tangents yields the new (red) control triangle. Note that the old (black) and new (red) curve segments overlap, as they describe the same curve.*

2.4 Approximating quadratic offset curves with quadratics

We choose to implement the desired approximate offset curve as a quadratic curve. Our algorithm begins by re-parameterizing the original curve. We construct a new control triangle that describes the initial curve, but for a segment which is symmetric with respect to the curve's axis. The resulting control triangle is then modified to achieve the desired conservative offset property.

2.4.1 Constructing the symmetric curve segment

To construct an axis-symmetric control triangle, we must first find the axis of the curve. Since our Bezier triangle describes a segment of a parabola, it has the property that the line l connecting the midpoint of the line p_0p_2 to the middle point p_1 is parallel to the axis a of the underlying parabola [Heath 1897]. Perpendicular to this line, we construct a chord that intersects the curve twice. Averaging the

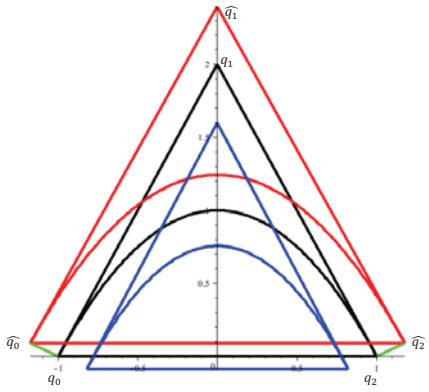

Figure 3: Constructing offset curves. *The black lines describe the initial control triangle and the quadratic curve segment it describes. The green lines show the normals used to displace the endpoints in the exterior direction by $d = 0.2$, yielding the red control triangle and correspoinding approximate exterior offset curve. The blue lines describe the same construction for an interior displacement of* 0.2.

intersection points yields a point on the axis, and thus the axis line and its intersection with the curve.

While an infinite number of symmetric control triangles can be constructed in this manner, all such triangles describe the same (original) curve. However, the choice of endpoints will be reflected in the offset curve/triangle constructed by our procedure. In order to preserve the initial tangent behavior for at least one endpoint of the original curve segment, we choose a chord that passes through either p_0 or p_2, preferring the endpoint that is more distant from p_1. This choice helps reduce numerical problems for asymmetric curve segments having one endpoint very close to p_1. Figure 2 shows a curve segment and its corresponding axis-aligned segment.

2.4.2 Constructing the offset curve

Given the extended, symmetric control triangle (with points q_0, q_1, and q_2), we wish to find a new triangle describing a conservative offset curve. Our procedure is similar to the initial step of Farin's offset algorithm [1989]. We choose a new control triangle such that

1. the original endpoints q_0 and q_2 are displaced outwards (for positive d) or inwards (for negative d), by a distance $|d|$ along the curve normal at the corresponding endpoint, yielding new endpoints $\hat{q_0}$ and $\hat{q_2}$, and
2. the original tangent vectors q_0q_1 and q_1q_2 are added to the corresponding new endpoints $\hat{q_0}$ and $\hat{q_2}$, yielding two lines which intersect at the new middle point $\hat{q_1}$.

All points on the resulting Bezier curve segment $\hat{q_0}\hat{q_2}$ lie at a distance $d' \geq d$ from their closest counterparts on the extended segment q_0q_2, and the same property holds for the original segment p_0p_2.

Figure 3 shows a symmetric control triangle, a pair of derived triangles describing approximate exterior and interior offset curves, and the curves themselves.

2.4.3 Motivating the symmetric construction

The initial step in our algorithm, namely the extension of the original control triangle to an axis-symmetric triangle describing the same curve, may appear unnecessary, suggesting that we could apply the displacement portion of the algorithm directly to the original triangle. Indeed, doing so would provide improved fidelity to the true offset curve at the initial endpoint not maintained by the symmetric construction (since the scaled-normal displacement would be applied there, rather than at a new point more distant from p_1).[2]

The problem is that applying the displacement operation to an asymmetric control triangle yields an offset curve whose axis differs from that of the original curve. As the offset is increased or decreased, the middle point (and thus the extreme curve point $F(0.5)$) moves outward (inward) along the line l connecting the midpoint to the center of the segment p_0p_2, which is parallel but not coincident to the axis. Since the inner and outer offset curves are thus skewed in opposite directions w.r.t. the axis, various anomalies, such as the outer offset curve failing to contain the inner offset curve, or the inner curve intersecting the curve segment, can occur.

These are not artificially-generated pathological cases; they appear even with small offsets to some curves in TrueType™ glyphs, typically when the control triangle is very narrow. While the symmetric form will typically generate less precise offset curves (because it fails to incorporate the tangent from one of the initial endpoints), it does not exhibit these anomalies.

2.4.4 Correctness

For a positive displacement d, we are able to show that, for any symmetric control triangle, this approach yields a conservative offset curve. Negative displacements are slightly more complex, as the true offset curve exhibits a singularity whenever the displacement exceeds the curve's radius of curvature. Once the normal displacement vectors from the endpoints (i.e., $\hat{q_0} - q_0$ or $\hat{q_2} - q_2$) reach the axis, there exists no point interior to the curve having distance $\delta \geq |d|$ to the portion of the curve segment lying on the same side of the axis, and the approximate offset curve degenerates to a single point.[3]

Given d sufficiently small that this does not occur, we can show that interior offset curves generated by our procedure are conservative. A detailed analysis can be found in the Appendix.

3 Experimental Results

We extracted 1,000 unique, non-degenerate quadratic curve segments from the Times New Roman TrueType font (beginning with the first ASCII character), and another 1,000 from the MS Mincho TrueType font (beginning with the first Japanese kana character). For each segment, we saved the bounds of the enclosing glyph's coordinate system, the three control points, described in terms of that coordinate system, and a flag indicating which side of the segment faces outward from the glyph contour to which it belongs. This information was used to drive our experiments. Where needed, precise distance values were computed by an iterative solver configured to return when the iteration-to-iteration difference in the distance value, measured in the curve's coordinate system, becomes less than 10^{-9}.

[2] The initial step of Farin's offset construction does precisely this. This initial imprecision is permissible because the algorithm recursively refines the problem until an arbitrary level of precision is achieved, whereas our algorithm is intended to return a single, conservative, solution.

[3] Extending the normals beyond the axis results in an inverted control triangle that does not have useful properties w.r.t. the original curve.

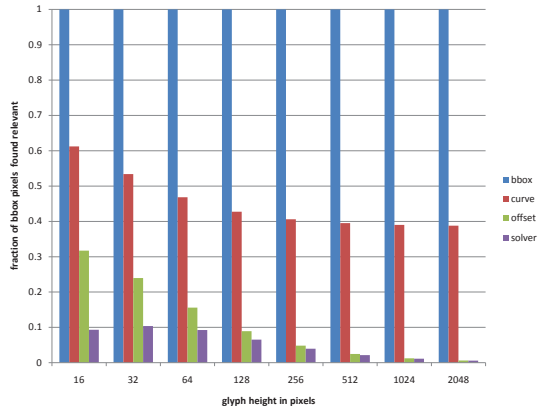

Figure 4: Antialiasing example. *The horizontal axis denotes glyph heights. The vertical axis shows the fraction of all sampled pixels found to be relevant (possibly within 1 pixel of the curve) via one of four tests: (1) lying inside the rectangular bounding box, (2) lying outside the curve, (3) lying inside the approximate offset curve, and (4) precise solver returning a distance of one pixel or less.*

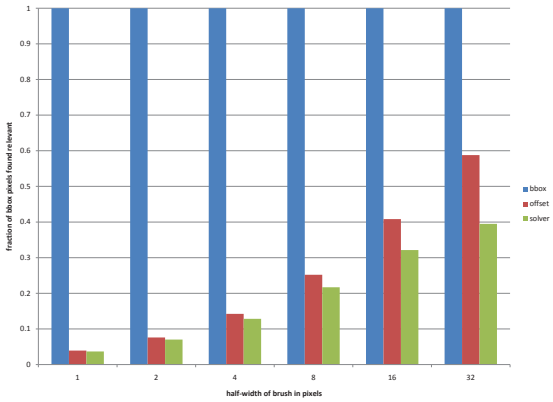

Figure 5: Brush example. *The horizontal axis denotes brush half-widths d in pixels. The vertical axis shows the fraction of all sampled pixels found to be within d pixels of the curve, as determined by one of three tests: (1) lying inside the rectangular bounding box, (2) lying between the exterior and interior approximate offset curves, and e, and (3) precise solver returning a distance $\leq d$.*

3.1 Antialiasing scenario

For outline-based fonts, antialiasing is performed on the outside of each curve segment. A distance-based transparency value is computed for all pixel positions lying within a single pixel width of the segment, on the appropriate side. The number of positions at which the precise distance computation is invoked can be limited as follows. We construct an approximate offset curve whose minimum distance value d corresponds to a 1-pixel displacement from the segment. Any point not lying between the segment and the offset curve is irrelevant to the antialiasing computation, and thus no precise distance computation need be performed for such points.

We analyze each curve segment in our corpus over a variety of glyph height values, expressed in screen pixels, ranging from 16 to 2048 pixels in height. Each height value defines a coordinate transformation between pixel and glyph coordinates. We map each segment's bounding rectangle to screen space, pad it by one pixel in each direction, and traverse it, querying the corresponding position in glyph space. All query points falling outside of the offset curve were found to have a point-curve distance greater than d; thus, the algorithm was shown to be (empirically) conservative.

Our acceleration technique was able to discharge 68–99% of queries. Figure 4 shows the fraction of in-bounding-box pixel queries found to be relevant by (a) the curve-side test, (b) containment between the approximate offset curve and the segment, and (c) our iterative solver. As the glyph height is increased, the one-pixel offset value in the screen coordinate system maps to successively smaller offset values in the glyph coordinate system, resulting in a more precise approximate offset curve. Unlike polygonal or hierarchical structures, which must grow as improved precision is required, our representation achieves improved precision while maintaining a constant representation size and query time.

3.2 Brush scenario

For a brushed path of width $2d$, sample points within d of a curve segment require a precise distance computation to determine the transverse parameter value. If antialiasing is to be performed, the relevant area increases by one pixel width on either side of the curve segment. We attempt to rule out irrelevant sample points by constructing two approximate offset curves with minimum distance values corresponding to the desired half-width, one on either side of the curve segment. Only points lying between the offset curves need be considered for further computation.

The data in Figure 5 describes the results of an experiment in which we scaled each curve segment in our corpus to a height or width of 128 pixels, depending on the segment's aspect ratio. For brush half-widths d varying from 1 to 32 pixels, we computed a tight bounding rectangle for the brush stroke, and evaluated the approximate and precise relevance conditions for each pixel position. We confirmed that all sampled points lying beyond either offset curve had point-curve distances greater than d. For $d \leq 8$, our acceleration technique discharges over 95% of irrelevant queries. As the brush width grows, the approximate offset curve becomes less precise, but remains useful, eliminating at least two thirds of the irrelevant queries in all cases.

4 Related Work

4.1 Bounding Regions

The idea of using bounding regions as an acceleration technique has a long history. As we mentioned in Section 2.2, most techniques either improve the precision of the lowest-level bounding representation elements, or build data structures to accelerate spatial queries returning sets of such elements. In an instance of the former approach, Kay and Kajiya [1986] bound objects with parallelopipeds instead of rectangular prisms.

The latter approach is more common: examples include general spatial data structures such as k-d trees [Bentley 1975] and r-trees [Guttman 1984], as well as graphics-rendering-specific instances [Rubin and Whitted 1980; Glassner 1988]. Recent works on representation of vector graphics [Nehab and Hoppe 2008; Qin et al. 2008] have used a single-level representation, the uniform grid.

Arvo and Kirk [1987] construct bounding descriptions not only of query targets (e.g., graphical objects) but also of groups of queries, allowing fast discharge of multiple queries at once. This idea applies to our offset scenario as well. For example, we can quickly show that a large number of query points are irrelevant for an-

tialiasing if all corner points of a rectangle enclosing these query points fall outside of a single-pixel-offset exterior approximate offset curve.

4.2 Offset Curves

The construction of precise planar offset curves has been the subject of much research [Elber et al. 1997; Anton et al. 2005; Tiller and Hanson 1984; Farin 1989]. Typically, the curve is repetitively subdivided until each resulting segment's offset curve can be represented with sufficient accuracy by a simpler construct such as a line segment or arc segment. Such recursive techniques can achieve an arbitrary level of precision, but share the disadvantages of the multi-polygonal bounding techniques described above.

Farin [1989] describes a subdivision-based procedure that represents offsets to segments of conic curves as conic curves. Our technique for extending the control triangle endpoints by the scaled curve normal is similar to his, but differs in that, since we are using only quadratics, we lose a degree of freedom (the w parameter) vs. that of conics.

4.3 Distance Computations

Loop and Blinn [Loop 2005] make use of the tesselation hardware in a GPU to efficiently evaluate the implicit definition of quadratic, conic, and cubic curve segments, thus obtaining an inclusion predicate, but not a distance function. Several iterative algorithms exist for computing the minimum distance vector from a point to a planar curve segment. Wang et al. [2002] specifically treats cubic curves, while Qin et al. [2008] partitions arbitrary curves into segments over which a binary subdivision-based search will find a single solution. When operating on quadratic curves, both of these algorithms could benefit from our acceleration technique. For the specific case of quadratic curves, Nehab and Hoppe [2008] present a closed-form approximation to the point-curve distance, which has been found to be accurate for points sufficiently near the curve.

5 Future Work

We plan to extend our technique to rational quadratic (conic) curves, exploiting their additional degree of freedom in fitting approximate offset curves. Additional avenues of potential exploration include trading endpoint precision for near-axis precision and the use of multiple approximate offset curves to model a single curve segment.

6 Conclusion

We have investigated the use of approximate offset curves as components of bounding mechanisms for regions of interest surrounding curve segments. In particular, we have shown that the space of coordinates relevant to bounded distance queries on quadratic Beizer curve segments can be usefully limited by in/out tests on other quadratic Bezier curves used as conservative, approximate offset curves. We have described a new technique for constructing such curves, presented an argument for its correctness, and demonstrated its utility on a corpus of real-world curves.

References

ANTON, F., EMIRIS, I., MOURRAIN, B., AND TEILLAUD, M. 2005. The offset to an algebraic curve and an application to conics. *ICCSA 2005 (LNCS 3480)*, 683–696.

ARVO, J., AND KIRK, D. 1987. Fast ray tracing by ray classification. *ACM SIGGGRAPH Computer Graphics 21*, 4, 55–64.

BENTLEY, J. L. 1975. Multidimensional binary search trees used for associative searcing. *Communications of the ACM 18*, 9, 509–517.

ELBER, G., LEE, I.-K., AND KIM, M.-S. 1997. Comparing offset curve approximation methods. *IEEE Computer Graphics and Applications 17*, 3, 62–71.

FARIN, G. 1989. Curvature continuity and offsets for piecewise conics. *ACM Transactions on Graphics 2*, 1, 89–99.

FARIN, G. 2002. *Curves and Surfaces for CAGD*. Morgan Kaufmann.

GLASSNER, A. S. 1988. Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications 8*, 2, 60–70.

GUTTMAN, A. 1984. R-trees: a dynamic index structure for spatial searching. In *ACM SIGMOD*, 47–57.

HEATH, T. L. 1897. *Archimedes: Works*. Cambridge University Press. 234–252.

KAY, T. L., AND KAJIYA, J. T. 1986. Ray tracing complex scenes. *ACM SIGGRAPH Computer Graphics 20*, 4, 269–278.

LOOP. 2005. Resolution independent curve rendering using programmable graphics hardware. *ACM Transactions on Graphics*, 1000–1009.

NEHAB, D., AND HOPPE, H. 2008. Random-access rendering of general vector graphics. *ACM Transactions on Graphics 27*, 5.

QIN, Z., MCCOOL, M. D., AND KAPLAN, C. 2008. Precise vector textures for real-time 3d graphics. *ACM Transactions on Graphics 27*, 5.

RUBIN, S. M., AND WHITTED, T. 1980. A 3-dimensional representation for fast rendering of complex scenes. In *ACM SIGGRAPH*, 110–116.

TILLER, W., AND HANSON, E. G. 1984. Offsets of two-dimensional profiles. *IEEE Computer Graphics and Applications 4*, 9, 36–46.

WANG, H., KEARNEY, J., AND ATKINSON, K. 2002. Robust and efficient computation of the closest point on a spline curve. *Curve and Surface Design*, 397–405.

Acknowledgements

The author would like to thank Jim Kajiya and Turner Whitted for discussions on bounding techniques and Andrew Glassner for drawing the examples in Figure 1.

Appendix

We wish to show that the approximate offset curves generated by our strategy are conservative. Without loss of generality, we examine the case of a curve segment described by a symmetric control triangle with control points $p_0 = (-1, 0)$, $p_1 = (0, h)$, and $p_2 = (1, 0)$, where $h \geq 0$.[4] This triangle describes a quadratic curve segment with parameterized form

$$p(t) = t^2(0, -2h) + t(2, 2h) + (-1, 0)$$

[4] Any control triangle can be reduced to this form via a sequence of affine translation, rotation, and scaling operations.

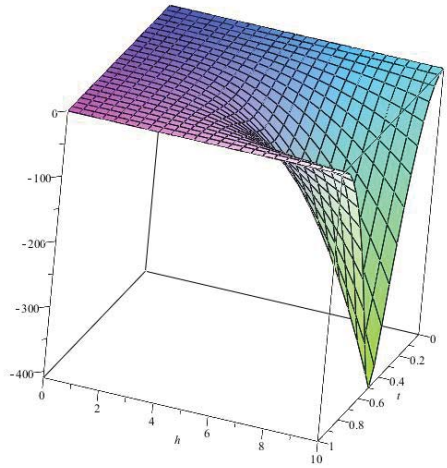

Figure 6: *B in h and t.*

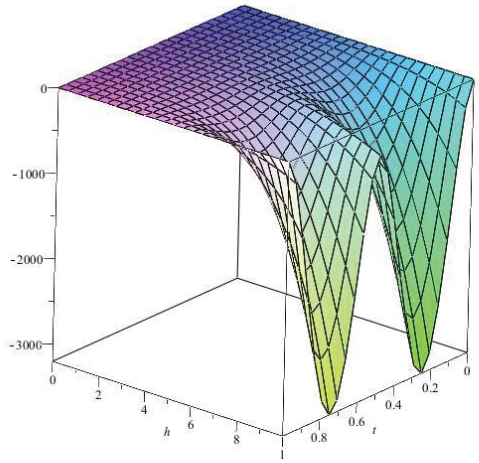

Figure 7: *C in h and t.*

where $0 \leq t \leq 1$ and implicit form[5]

$$c(p) = b_1(p)^2 - 4b_0(p)b_2(p)$$

where the $b_i(p)$ are the barycentric coordinates of point p with respect to the control triangle points p_i. The term $c(p)$ will be positive for points p outside the curve, zero for points on the curve, and negative for points inside the curve.

We can construct a precise offset curve in parameterized form by adding an appropriately scaled normal vector at each point. Given an offset distance $o \geq 0$, the the exterior and interior offset curve functions are

$$q(t) = p(t) + n(t)$$

and

$$r(t) = p(t) - n(t)$$

.

where

$$n(t) = \left(\frac{-(p'(t)_y}{\| p'(t) \|}, \frac{p'(t)_x}{\| p'(t) \|}\right) o$$

is the scaled normal vector field along the curve.

Our procedure for constructing an approximate offset curve builds a control triangle whose endpoints are precise; i.e., for the exterior case $q_0 = q(0)$ and $q_2 = q(1)$. The middle point q_1 lies at the intersection of the tangent line $q(0) + q'(0)$ and the y axis. The internal case is similar. By substituting the new control points into the implicit formula $c(p)$, we obtain functions $c_q(t)$ and $c_r(t)$ that compute whether or not the approximate offset curve encloses the point lying on the true offset curve at parameter value t.

We begin by analyzing the exterior offset case. Recall that we have assumed

[5]This is the implicit form for conics given in [Farin 2002, p. 216], specialized for the parabolic case.

- $h > 0$ (definition of canonical triangle),
- $o \geq 0$ (we're handling external and internal offsets explicitly),
- $0 \leq t \leq 1$ (we only care about distances to the curve segment, not the entire curve)

The function $c_q(t)$ represents the inclusion of the true exterior offset curve within the approximate exterior offset curve, and thus should never be positive. We can factor it as

$$c_q(t) = A(t)(oB(t) + C(t))$$

where

$$A(t) = \frac{2\,o}{h\left(\sqrt{1+h^2} + oh\right)^2 (1 + h^2(2t-1)^2)}$$

$$B(t) = \begin{pmatrix} -h^3 + h\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2}\sqrt{1+h^2} \\ -2\,h^3t^2 + 2\,h^3t - h \end{pmatrix}$$

$$C(t) = \begin{pmatrix} 2\,h^2\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2} \\ +h^4\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2} \\ +\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2} \\ +4\,t^2h^2\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2} \\ -4\,th^2\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2} \\ -2\,h^2\sqrt{1+h^2} - 14\,h^4\sqrt{1+h^2}t^2 \\ +6\,h^4\sqrt{1+h^2}t - 6\sqrt{1+h^2}t^2h^2 \\ +6\sqrt{1+h^2}th^2 + 16\sqrt{1+h^2}t^3h^4 \\ +4\,t^2h^4\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2} \\ -4\,th^4\sqrt{1 + 4\,t^2h^2 - 4\,th^2 + h^2} \\ -8\sqrt{1+h^2}t^4h^4 - \sqrt{1+h^2} - h^4\sqrt{1+h^2} \end{pmatrix}$$

The term $A(t)$ is clearly zero or positive, so we can ignore it for the remainder of our analysis. $B(t)$, and thus $oB(t)$, can be shown analytically to be zero or negative for all relevant h. A plot of B in h and t (see Figure 6) shows that the imprecision of the approximate offset curve, as a function of the offset o, behaves as we might

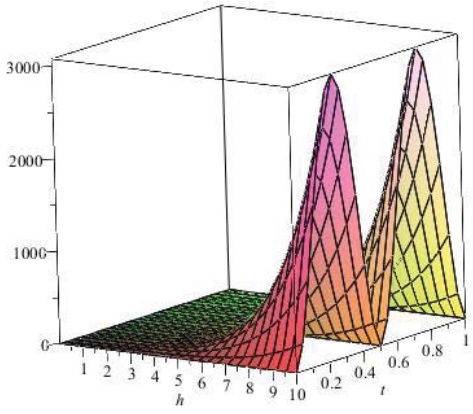

Figure 8: $\hat{D}$ *in* h *and* t.

expect. At the endpoints, the approximate curve is perfect by construction, so the offset value o has no effect ($B(t) = 0$), while the offset-induced imprecision is worst at the axis crossing ($t = .5$) and increases in magnitude as h is increased.

While most of the terms comprising $C(t)$ can be grouped into pairs having a negative sum, the remaining unpaired terms, $h\sqrt{1+h^2}(16t^3 - 14t^2 + 6t)$, always sum to a zero or positive value. Thus, we must consider $C(t)$ as a whole. In Figure 7, we plot C for $0 \leq t \leq 1$ and $0 \leq h \leq 10$.

For this range of h, the value of $C(t)$ is always zero or negative. We can be assured that the plot is not missing any high-frequency components in t because the derivative with respect to t has five zeros, all of which lie between 0 and 1, and we can see three zeros and two inflection points in the plot, so nothing's gone missing. What we don't know is whether increasing the value of h might somehow cause the value to exceed 0. By plotting the first and second derivatives of C with respect to h for $0 \leq t \leq 1$ and $1 \leq h \leq 10^6$ and observing all values to be zero or negative, we can feel confident that $C(t)$ will always have a non-positive value.

Finally, given that both $B(t)$ and $C(t)$ are known to be non-positive, and are only scaled by positive quantities, we can conclude that $c_q(t)$ is zero or negative for all relevant values of h, o, and t, indicating that any point on the precise exterior offset curve segment will be contained by our exterior approximate offset curve.

We now turn our attention to case of interior offsets, and examine the evaluation function $c_r(t)$. Before we begin, we need to restrict the range of the offset value o in our analysis. As we noted earlier, the true offset curve exhibits a singularity whenever the displacement exceeds the curve's radius of curvature; making the offset any larger would cause the the offset vectors to cross the axis (and one another), yielding a downward-facing control triangle and a meaningless approximate inner offset curve. In such situations, our implementation returns a constant-valued characteristic function that always returns a positive value, indicating that no points are enclosed by the inner approximate offset curve. For purposes of analysis, we model this by requiring $r(0)_x \leq 0$.

Like its counterpart, the external evaluation function $c_q(t)$, the internal evaluation function $c_r(t)$ can be factored into a scaled sum

$$c_r(t) = A(t)D(t)$$

where

$$D(t) = oB(t) - C(t)$$

and $A(t)$, $B(t)$, and $C(t)$ are as before. From above, we know that $C(t)$ is non-positive, so its contribution to $c_r(t)$ must be non-negative. As $B(t)$ is non-positive, we are unable to reason about the sign of $D(t)$ merely from the signs of its components. A straightforward graphical analysis would require a four-dimensional plot, as all of h, o, and t are in play at the same time. We can eliminate the parameter o by noting that the largest o (and thus the most negative product $oB(t)$) permitted by the restriction $r(0)_x \leq 0$ is obtained when $r(0)_x = 0$. Solving for o in this formula (yielding $o = \frac{\sqrt{1+h^2}}{h}$) and substituting into $D(t)$ gives us a two-parameter formula for $D(t)$ minimized over all o, namely

$$\hat{D}(t) = \begin{pmatrix} (8\,h^4\,t^4 - 16\,h^4\,t^3 + 4\,h^2\,t^2 + 14\,h^4\,t^2 \\ -4\,h^2\,t - 6\,h^4\,t + h^2 + h^4)\sqrt{1+h^2} \\ +(-4\,t^2\,h^2 - 4\,t^2\,h^4 + 4\,h^2\,t + 4\,h^4\,t - h^2 - h^4) \\ \sqrt{1 + 4\,h^2\,t^2 - h^2\,t + h^2} \end{pmatrix}$$

which is plotted in Figure 8. As we did with $C(t)$ above, we can examine $\hat{D}(t)$'s derivatives with respect to t and h and conclude that the plotted region accurately represents the function's sign (in this case, non-negative) over its entire domain. Since $\hat{D}(t)$ represents the worst-case behavior for $D(t)$, we have shown that, for appropriately restricted parameter values, $c_r(t)$ is non-negative. Thus, we are able to claim that any point on the precise interior offset curve segment will lie outside of our approximate interior offset curve.