[3] C. Bajaj, C. Hoffmann, R. Lynch, and J. Hopcroft. Tracing surface intersections. *Computer Aided Geometric Design*, 5:285–307, 1988.

[4] C. Bajaj and M.-S. Kim. Convex hulls of objects bounded by algebraic curves. *Algorithmica*, 6:353–553, 1992.

[5] J.-D. Boissonnat, A. Cerezo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension D. Technical report, INRIA, June 1992.

[6] J. Canny. *The Complexity of Robot Motion Planning.* MIT Press, 1988.

[7] D. P. Dobkin and D. L. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5(3):421–457, 1990.

[8] R. Farouki. The characterization of parametric surface sections. *Comp. Vision, Graphics, and Image Proces.*, 33:209–236, 1986.

[9] J. Johnstone and C. Bajaj. Sorting points along an algebraic curve. *SIAM J. Comput.*, 19(5):925–967, 1990.

[10] J. Koenderink. *Solid Shape.* MIT Press, Cambridge, MA, 1990.

[11] D. Kriegman and J. Ponce. Geometric modelling for computer vision. In *SPIE Conference on Curves and Surfaces in Computer Vision and Graphics II*, Boston, MA, 1991.

[12] D. Kriegman and J. Ponce. A new curve tracing algorithm and some applications. In P. Laurent, A. L. Méhauté, and L. Schumaker, editors, *Curves and Surfaces*, pages 267–170. Academic Press, New York, 1991.

[13] D. Manocha and J. Canny. Algorithm for implicitizing rational parametric surfaces. In *IMA Conf. on Mathematics of Surfaces*, Bath, 1990. To appear in Computer Aided Geometric Design.

[14] M. Marden. *Geometry of Polynomials*, volume 3 of *Mathematical Surveys.* American Mathematical Society, Providence, RI, 1966.

[15] A. Morgan. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems.* Prentice Hall, Englewood Cliffs, 1987.

[16] A. P. Morgan. A transformation to avoid solutions at infinity for polynomial systems. *Applied Mathematics and Computation*, 18:77–86, 1986.

[17] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[18] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[19] A. A. Schäffer and C. J. V. Wyk. Convex hulls of piecewise-smooth jordan curves. *J. Algorithms*, 8:66–94, 1987.

[20] J. Schwartz and M. Sharir. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. In J. Schwartz, M. Sharir, and J. Hopcroft, editors, *Planning, Geometry and Complexity of Robot Motion*, pages 51–96. Ablex Publishing Corp., Norwood, NJ, 1987.

[21] T. W. Sederberg, D. Anderson, and R. N. Goldman. Implicit representation of parametric curves and surfaces. *Comp. Vision, Graphics, and Image Proces.*, 28:72–84, 1984.

[22] G. Taubin. Rasterizing implicit cruves by space subdivision. Technical Report RC 17913, IBM Computer Science, 1992.

[23] R. Walker. *Algebraic Curves.* Princeton University Press, 1950.

we have implemented homotopy continuation on networks of SUN SPARC Stations communicating via Ethernet, networks of INMOS Transputers, and INTEL Hypercubes. In practice, this allows us to routinely solve systems with a few thousand roots, a task that requires a few hours using a dozen SPARC Stations. The classification algorithm has also been implemented and is used as part of convex hull procedure.

The algorithm has been applied to a number of algebraic curves as shown in figures 5–7. The classical curve called the limaçon of Pascal (figure 5) contains one singular point and only one bitangent line. Figure 6 shows an instance of the lemniscate of Bernoulli with its characteristic figure eight shape. There are two bitangent lines in this case. Finally, figure 6 shows a quartic curve with two singular points (one is a simple crossing while the other is a tacnode) and its convex hull. Computing the convex hull of each of these quartic curves required approximately twenty minutes, most of which was spent solving the bitangent system (3).

4 Discussion

In this paper we have presented an algorithm for computing the convex hull of an irreducible algebraic curve. The curve may be composed of multiple components and may include singular points. In addition, we have presented an algorithm for classifying points with respect to the convex hull.

In many applications, curves are represented parametrically which greatly simplifies convex hull construction. For a closed curve, a map $p : S^1 \rightarrow \mathbb{R}^2$ is available. The bitangent lines can be computed by solving a system of two equations in two unknowns $(s_1, s_2 \in S^1)$; for each bitangent, s_1 and s_2 bound an interval of the circle S^1. Hull construction amounts to completely partitioning S^1 into a set of non-overlapping intervals corresponding either to a bitangent line whose end points are the image of interval bounds or to an interval of the original curve.

The algorithm can be trivially extended to a collection of n irreducible algebraic curves $\mathcal{C}_i$ given by $f_i(x, y) = 0$. While such a collection can be represented as an algebraic curve

$$f(x, y) = \Pi_{i=1}^{n} f_i(x, y) = 0,$$

and the presented algorithm could be applied, it is far more efficient to explicitly consider the collection of separate curves. Bitangents of each $\mathcal{C}_i$ are found individually followed by the bitangents between two curves $\mathcal{C}_i$ and $\mathcal{C}_j, i \neq j$. Extremal points of each $\mathcal{C}_i$ can be computed. Additionally, singular points are formed by the intersection of two curves which are easily found by solving a system of polynomial equations. The rest of the algorithm would basically follow that of section 2.

We are also considering algorithms for extending the ideas presented here to bounded algebraic surfaces in $\mathbb{R}^3$. As noted by Koenderink [10], the convex hull is composed of three types of surface points: Planes that have second order contact with the surface at three points, developable surfaces where each ruling has second order contact at two surface points, and portions of the original surface. The planes are found by solving a square system of polynomial equations while the ruled surface is found by tracing a curve [12] defined by five equations in six unknowns. The original surface can be decomposed by considering the critical points of the projection onto a plane using ideas discussed in [11]. These points form a curve in $\mathbb{R}^3$ which can be similarly traced. Again, we must face the issue of finding an appropriate output representation that is useful both for rendering and for geometric algorithms.

5 Acknowledgments

This work was supported in part by the National Science Foundation under Grant IRI-9015749.

References

[1] D. Arnon, G. Collins, and S. McCallum. Cylindrical algebraic decomposition I and II. *SIAM J. Comput.*, 13(4):865–889, November 1984.

[2] D. S. Arnon. Topologically reliable display of algebraic curves. *Computer Graphics*, 17(3):219–227, July 1983.

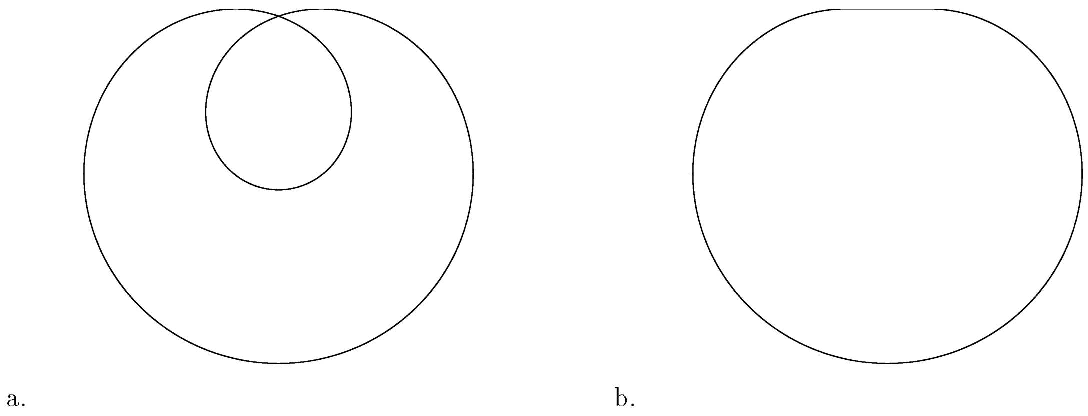

Figure 5: a. A singular quartic curve given by $(x^2+y^2+6y)^2-4(x^2+y^2)=0$. b. Its convex hull.

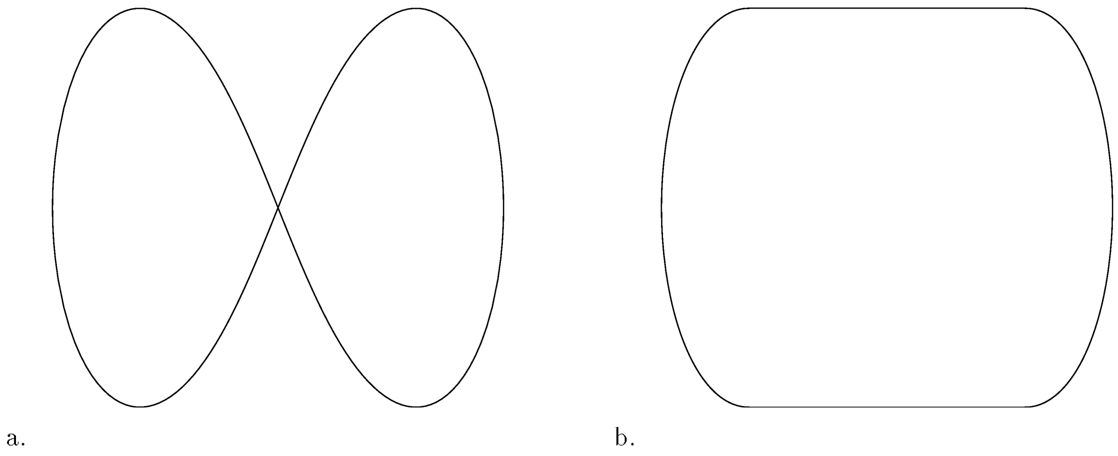

Figure 6: a. A singular quartic curve given by $(x^2+y^2)^2-4(x^2-y^2)=0$. b. Its convex hull.

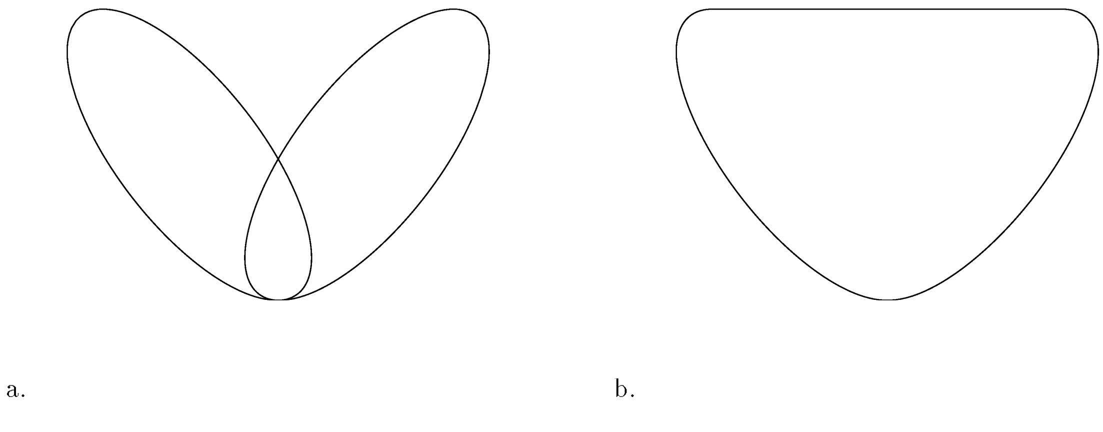

Figure 7: a. A singular quartic curve given by $y^4-2y^3+y^2-3x^2y+2x^4=0$. b. Its convex hull.

of the interval. These points are found by solving the univariate polynomial

$$f(x_s, y) = 0 \tag{6}$$

for y using either continuation or some other polynomial solution technique [18].

Some of these sample points will lie on the convex hull $\mathcal{H}$, and others will be inside $\mathcal{H}$. The classification method of section 2.3 can be used to determine which sample points lie on the convex hull. Alternatively, only sample points with maximal or minimal y-coordinate can be on $\mathcal{H}$, so the Sturm sequence test is unnecessary; comparison against the bitangents $\tilde{\mathcal{L}}$ is still required. Along with the interval boundaries, this yields a representation of the set $\tilde{\mathcal{B}}$ of curve branches on the convex hull. The degree of (6) is d, and so there may be d sample points of $\mathcal{C}$ per interval, but at most two sample points per interval may be on $\mathcal{H}$.

2.4.3 Tracing

To actually render the curve branches, the curve is traced using a classical prediction/correction approach based on a Taylor expansion of $f(x, y)$ [3, 8, 12]. From a first order expansion of f (higher order expansions could be used [3]), we have

$$dy = \frac{(\partial f/\partial x)}{(\partial f/\partial y)} dx. \tag{7}$$

From a point (x, y) and given a step size dx in the x direction, the predicted next point is $(x + dx, y + dy)$. Note that $\partial f/\partial y$ is always non-zero since the interval is guaranteed to be extremum and singularity free by step one of the curve decomposition algorithm. The correction step simply uses Newton iterations to converge back to the curve from the predicted point $(x + dx, y + dy)$ [18]. For a fixed value of x, steps in y are taken according to

$$dy = -\frac{f(x, y)}{\partial f/\partial y}. \tag{8}$$

After tracing the curve branches from the sample points, figure 4 shows the convex hull composed of curve branches $\tilde{\mathcal{B}}$ and four bitangent line segments.

2.5 Complexity and Size

We now consider the time complexity of the algorithm and the size of the resulting convex hull representation for an algebraic curve $\mathcal{C}$ of degree d. As mentioned in section 2.1, the computational complexity of solving polynomial systems by homotopy continuation is unknown, so the following is based on the use of resultant methods. To compute the bitangent set $\mathcal{L}$, (3) is solved which requires time $O(d^{16})$ and yield $O(d^4)$ bitangents. To determine $\tilde{\mathcal{L}}$ requires computing the hull of $\mathcal{L}$ and time $O(d^4 \log d)$. From section 2.3, classifying the $O(d^4)$ endpoints in $\mathcal{L}$ against $\mathcal{C}$ requires $O(d^5 \log^2 d)$ operations. Since all bitangent endpoints may be on the hull of $\mathcal{C}$, the size of $\tilde{\mathcal{L}}$ is $O(d^4)$, and the time to compute $\tilde{\mathcal{L}}$ is (d^{16}).

When decomposing the curve $\mathcal{C}$ as in section 2.4, there are $O(d^2)$ possible extrema and singular points which are found in $O(d^6 \log d)$ operations. The merged extrema and bitangent endpoints yields $O(d^4)$ intervals which can be sorted in $O(d^4 \log d)$ time. For each interval, the sample points are found by solving a degree d polynomial requiring a total of $O(d^7 \log d)$ operations. As mentioned in section 2.4.2, at most two samples per interval will actually be on the hull.

Therefore, the size of the convex hull representation of a degree d algebraic curve is $O(d^4)$ while the computation time is dominated by computing the bitangents which is $O(d^{16})$.

3 Implementation and Examples

The algorithm presented here has been implemented mostly in Common Lisp. Given an equation $f(x, y)$, the appropriate polynomial systems are formed using the REDUCE computer algebra package. To solve these systems,

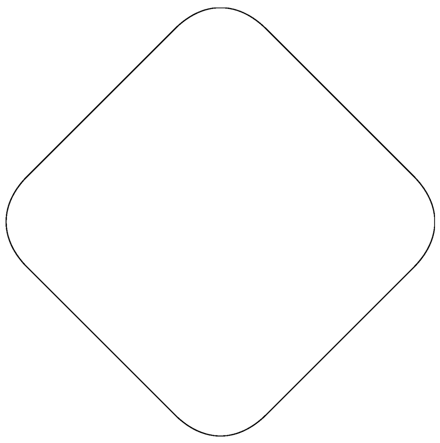
Figure 4: The convex hull of the curve of Fig. 1.

The algorithm for decomposing the curve $\mathcal{C}$ is divided into 4 steps which are illustrated in figure 3.

1. Compute the extremal and singular points of $\mathcal{C}$ in the x direction.
2. Take the union of the extremal and singular points from step 1 with the endpoints of the bitangents $\tilde{\mathcal{L}}$, and sort them by their x-coordinate; the x-coordinate of successive points will delimit an interval.
3. For each interval of the x axis, compute the intersection of the curve with the line of constant x at the interval midpoint to obtain one sample for each curve branch.
4. For rendering, march numerically from the sample points found in step 3 to the interval bounds found in step 2 by predicting new points through Taylor expansion and correcting them through Newton iterations.

Because of step one, there will be at least one sample on every curve component, and each open interval will not contain singular or extremal points; thus, the branch can be parameterized by x, and it can be traced by evaluating the following integral equation from the sample point $\mathbf{s} = (x_s, y_s)$ to an interval limit ($x+$ or $x-$):

$$y(x) = \int_{x_s}^{x+} \left(\frac{dy}{dx}\right) dx + y_s. \tag{4}$$

We now explain these steps in more detail.

2.4.1 Extremal and Singular Points

The extremal and singular points of $\mathcal{C}$ are found by solving the following system of polynomial equations

$$\begin{cases} f(x,y) = 0 \\ \frac{\partial}{\partial y} f(x,y) = 0 \end{cases} \tag{5}$$

for (x, y) using homotopy continuation. Note that the curve normal is undefined at singular points, so they are characterized by (5) and $\frac{\partial f}{\partial x} = 0$. The singular points are a subset of the roots found by solving (5). The total degree of this system and consequently the maximum number of extremal and singular points is $d(d-1)$.

2.4.2 Sample Points

After merging extremal and singular points with the bitangent endpoints and sorting them by their x-coordinate, the sample points are given by the intersection of $\mathcal{C}$ with the line of constant x (vertical) through the midpoint x_s

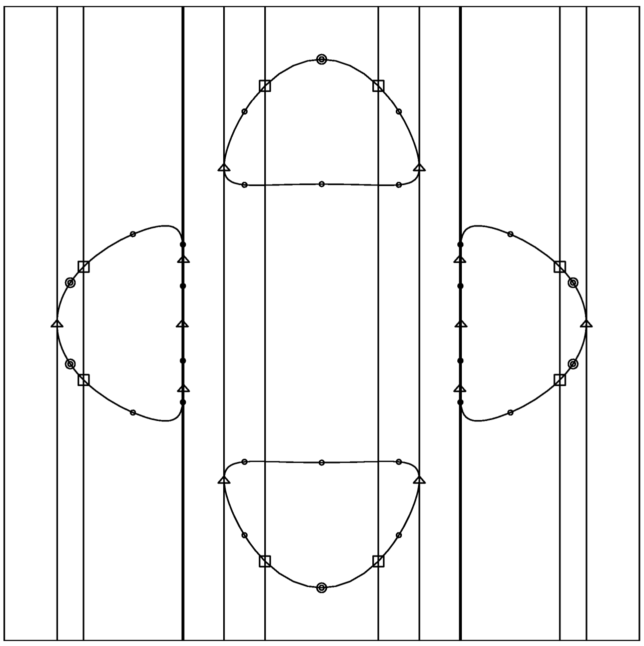

Figure 3: The decomposition of the curve of Fig. 1. Extremal points are indicated by triangles (there are no singular points), and the endpoints of the bitangents are marked with squares. The vertical lines indicate the interval boundaries, and the sample points of each branch are marked with a circle; the large circles lie on the convex hull while the small ones are within the hull.

Since every point on the hull is either on the curve $\mathcal{C}$ or on the set of bitangent segments $\tilde{\mathcal{L}}$ (i.e., $\mathcal{H} \subseteq \mathcal{C} \cup \tilde{\mathcal{L}} \subseteq \mathcal{C} \cup \mathcal{L}$), every line through $\mathbf{p}$ intersects $\mathcal{C} \cup \tilde{\mathcal{L}}$ in *at least* one point on each side of $\mathbf{p}$. Since this is true for every line, any line will do, and so, without loss of generality, we choose the "vertical" line through $\mathbf{p} = (x_0, y_0)$ given by $x = x_0$. We now detail how to compute the number of intersections of this line first with the set of bitangents $\tilde{\mathcal{L}}$ and then with the curve $\mathcal{C}$.

Each intersection with $\tilde{\mathcal{L}}$ can be determined in $O(\log n)$ time. Briefly, since the segments of $\tilde{\mathcal{L}}$ are an ordered subset of a convex polygon, $\tilde{\mathcal{L}}$ can be split into ordered upper and lower sequences by finding the two extremal endpoints in the x direction. Within these sequences, segments define non-overlapping intervals of x. So, given a point $\mathbf{p} = (x_0, y_0)$, the segment in $\tilde{\mathcal{L}}$ (if it exists) that intersects the vertical line $x = x_0$ can be found by binary search according to x-coordinate of the endpoints in $O(\log n)$ time.

The number of intersections on each side of $\mathbf{p}$ of the line $x = x_0$ with the curve $\mathcal{C}$ could be found by explicitly computing intersection points (e.g. solving the univariate polynomial $f(x_0, y) = g(y) = 0$ for y). However, since we are not interested in the actual intersection points, a more efficient method is to simply determine the number of roots of $g(y) = 0$ between $[-\infty, y_0]$ and $[y_0, \infty]$ using Sturm sequences as described in section 2.1. Given $\mathbf{p}$, the time to compute $g(y) = 0$ from $f(x, y) = 0$ is $O(d^2)$ (see [4]), and the Sturm sequence is then computed and evaluated in $O(d \log^2 d)$ where d is the degree of f.

2.4 Decomposing the curve

The next step in the algorithm is to decompose the curve $\mathcal{C}$ into a set $\mathcal{B}$ of regular curve branches where the curve can be parameterized by the x-coordinate over some interval of the x axis (i.e. there exists a bijective map $x \mapsto (x, h(x))$ though no analytic form for h need exist); in addition, the branches do not contain the endpoints of the bitangents $\tilde{\mathcal{L}}$ except perhaps as an endpoint. This approach is motivated by the cylindrical algebraic decomposition of the curve [1, 2], and is based on a curve-tracing algorithm [12]. Since there is a bijective transformation between the branch and the x axis, it will be represented by a sample point $\mathbf{s} \in \mathbb{R}^2$ and an interval $(x-, x+)$ of the x axis. Additional points on the branch can be found by tracing the curve from $\mathbf{s}$ to the interval bounds.

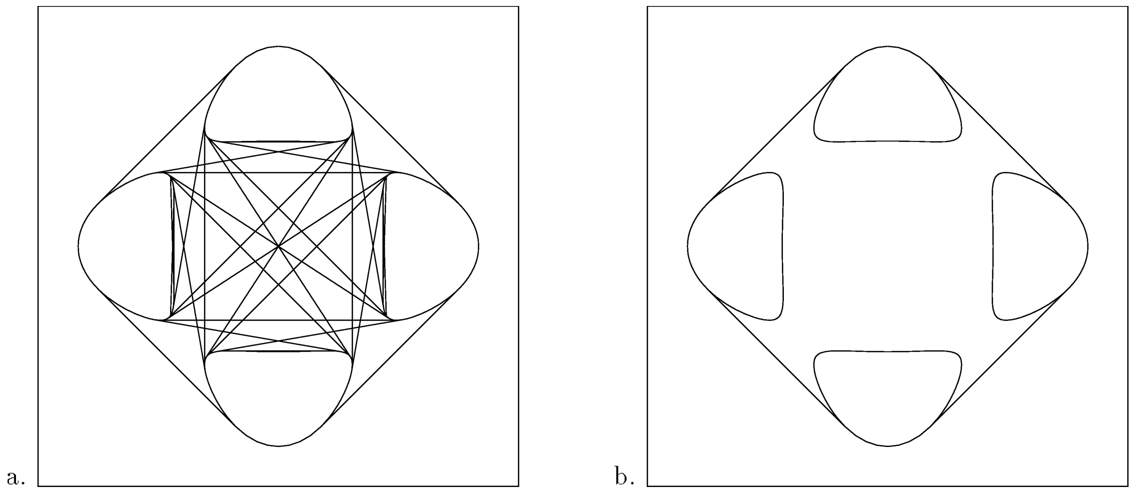

Figure 2: a. All bitangents $\mathcal{L}$ of the curve in Fig. 1. b. The bitangents $\tilde{\mathcal{L}}$ on the convex hull.

$$\begin{cases} f(\mathbf{p}_1) = 0 \\ f(\mathbf{p}_2) = 0 \\ \nabla f(\mathbf{p}_1) \cdot (\mathbf{p}_1 - \mathbf{p}_2) = 0 \\ \nabla f(\mathbf{p}_2) \cdot (\mathbf{p}_1 - \mathbf{p}_2) = 0 \end{cases} \tag{3}$$

which simply expressed that the endpoints lie on the curve, and the line is perpendicular to the curve normal. A couple of points should be noted. Segments with one or two singular endpoints are also characterized by (3) since the gradient will vanish at these points. As well as isolated solutions, the system (3) contains an improper component given by $f(\mathbf{p}_1) = 0$ and $\mathbf{p}_1 = \mathbf{p}_2$. An upper bound for the size of $\mathcal{L}$ is d^4 as given by the total degree of (3). Figure 2.a shows the curve of figure 1 and the bitangent lines.

The set $\mathcal{L}$ is a superset of those segments that actually lie on the convex hull since the system (3) only characterizes the local geometry of the endpoints. To determine the set $\tilde{\mathcal{L}}$ of bitangents on the hull, we follow a two step procedure. First, we note that a necessary (but not sufficient) condition for an endpoint $\mathbf{p}$ of a segment $l \in \mathcal{L}$ to be on the convex hull of $\mathcal{C}$, is that $\mathbf{p}$ is on the convex hull of $\mathcal{L}$. This is not sufficient because even though $\mathbf{p}$ may be on the hull of $\mathcal{L}$, it may be inside $\mathcal{C}$. Thus, the first step is to construct the convex hull of the endpoints of $\mathcal{L}$ and retain those segments with both endpoints being on this hull. This requires $O(n \log n)$ time where n is the number of segments [17]. The second step is to remove those segments which have an endpoint within the curve using the method described in section 2.3. Note that when constructing the hull of $\mathcal{L}$, the vertices are ordered to form a polygon, and this ordering is retained in $\tilde{\mathcal{L}}$. Figure 2.b shows the curve and the set $\tilde{\mathcal{L}}$.

2.3 Point Classification

To determine which bitangents are part of the hull and for subsequent steps, an algorithms is required to classify a point $\mathbf{p} = (x_0, y_0)$ as being inside, on, or outside of the convex hull. As noted in the introduction, this is an easy task when the hull is a polygon, but is more difficult for the hull of an algebraic curve. A line divides $\mathbb{R}^2$ into halfspaces according to the sign of the line equation, and so the sign of the equation evaluated a point can be used for classification. In contrast, an algebraic curve may divide $\mathbb{R}^2$ into a number of disjoint regions, and so the sign is not sufficient for classifying a point as being in a particular region. For example, the curve defined by $f(x,y) = (x^2+y^2-1)(x^2+y^2-4) = 0$ (two concentric circles of radii 1 and 2) decomposes $\mathbb{R}^2$ into three regions; $f < 0$ corresponds to an open annulus while $f > 0$ corresponds to two open regions (points within the circle of radius 1 and points outside of circle of radius 2). Thus, for algebraic curves, the sign of f is not sufficient for point classification. By the definition of convexity, a point $\mathbf{p}$ lies within the convex hull if the intersection of every line passing through $\mathbf{p}$ transversally intersects the hull boundary in one point on each side of $\mathbf{p}$. However, point classification can be accomplished without actually having a representation of the hull boundary.

3. Decompose $\mathcal{C}$ into a set $\mathcal{B}$ of closed curve branches. Only the endpoints of each branch may be singular, extremal in some direction, or an endpoint of a segment in $\tilde{\mathcal{L}}$.

4. Each branch in $\mathcal{B}$ will lie entirely on or inside of the hull, so remove those branches that are within the hull to create a new set $\tilde{\mathcal{B}}$.

The convex hull is given by the union of the sets $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{B}}$. After summarizing some key computations reported elsewhere, details of the above steps will be presented.

2.1 Preliminaries

Since the curves considered here are algebraic, a few fundamental algorithms for manipulating and solving polynomial equations are employed when computing the convex hull; this includes finding the roots of a system of polynomial equations and determining the number of real roots of a univariate polynomial within an interval.

Consider a system of n polynomial equations p_i in n unknowns x_j, denoted by $\mathbf{p}(\mathbf{x}) = \mathbf{0}$, with $\mathbf{p} = (p_1, \ldots, p_n)^T$ and $\mathbf{x} = (x_1, \ldots, x_n)^T$. There are two competing techniques for solving such systems. The Sylvester resultant [23] or the u-resultant [6] can be used to reduce the system to a univariate equation which is then solved. Because of numerical issues when computing the resultant, exact arithmetic must be used which renders this approach impractical. Instead, we use the homotopy continuation method to solve this system [15]. The principle of the method is as follows. Let $\mathbf{q}(\mathbf{x}) = \mathbf{0}$ be another system of polynomial equations with the same total degree as $\mathbf{p}(\mathbf{x}) = \mathbf{0}$, but with known solutions. A homotopy, parameterized by $t \in [0, 1]$, can be defined between the two systems by:

$$(1 - t)\mathbf{q}(\mathbf{x}) + t\mathbf{p}(\mathbf{x}) = \mathbf{0}. \qquad (2)$$

The solutions of the target system are found by tracing a curve (i.e. integrating a differential equation) defined in $\mathbb{R}^{n+1}$ by these equations from $t = 0$ to $t = 1$. The method of actually tracing this and other curves will be discussed more fully in section 2.4.3. In this case, the initial conditions of the differential equation are the known solutions of $\mathbf{q}(\mathbf{x}) = \mathbf{0}$ at $t = 0$. It can be shown [15] that for almost any choice of $\mathbf{q}$, the curve has no extrema, bifurcations, or singularities. An example of $\mathbf{q} = (q_1, \ldots, q_n)$ is given by $q_i(\mathbf{x}) = x_i^{deg(p_i)} - r_i$ where $deg(p_i)$ is the degree of equation p_i, and r_i is a random complex number. Note that this algorithm returns both the real and complex roots of $\mathbf{p}$, though only the real ones are relevant here.

According to Bezout's theorem, the maximum number of (real) isolated roots is given by the total degree of p (i.e. $\Pi_{i=1}^{n} deg(p_i)$), even in the presence of the solutions of excess dimension [16]. Unfortunately, we are unaware of any time complexity results concerning this approach, and will use the time complexity analysis for the resultant-based technique. As shown in [4] using the RAM model of computation with arithmetic operations requiring unit time cost, the time to solve a univariate polynomial of degree d is $O(d^3 \log d)$ [20], the time to solve two polynomial equations of degree d is $O(d^6 \log d)$, and the time to solve four polynomial equations of degree d is $O(d^{16})$.

The other fundamental algorithm of interest is determining the number of real roots of a univariate polynomial $f(x) = \sum_{i=0}^{d} c_i x^i$ of degree d within an interval $x \in (a, b)$. While the number of roots could certainly be determined by solving f and then counting the roots in (a, b), a much faster method is to use a Sturm sequence [14]. Following the discussion in [20], the Sturm sequence of f is a sequence of polynomials $\{f_i\}, i = 0..d$ which can be recursively constructed starting from $f_0 = f$ and $f_1 = f'$; f_i is the remainder of dividing $-f_{i-2}$ by f_{i-1}. Since the degree of f_i is less than the degree of f_{i-1} by construction, this sequence terminates with a constant. The number of roots of f within the interval (a, b) is then given by $S(a) - S(b)$ where $S(x)$ is the number of sign changes in the Sturm sequence evaluated at x. Schwartz and Sharir present a method using a fast GCD procedure for determining and evaluating the Sturm sequence in $O(d \log^2 d)$ time where d is the degree of the polynomial.

2.2 Computing Bitangents

The first step in the algorithm is computing the set of line segments $\mathcal{L}$ on the convex hull whose endpoints $\mathbf{p}_1, \mathbf{p}_2$ are tangent to $\mathcal{C}$ or are singular points of $\mathcal{C}$; with abuse, we will sometimes refer to this set as the bitangents of $\mathcal{C}$. The normal to the curve at a point $\mathbf{p}$ is given by $\nabla f(\mathbf{p})$, the gradient of f. $\mathbf{p}$ is singular (e.g. the tangent is undefined) if $\nabla f(\mathbf{p}) = 0$. Thus, the set of bitangent lines can be characterized by the system of equations

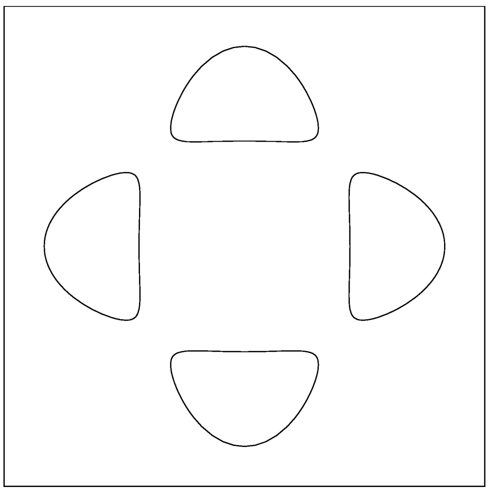

Figure 1: An algebraic curve with four components given by: $f(x,y) = 4y^4 + 17x^2y^2 - 20y^2 + 4x^4 - 20x^2 + 17$.

Assuming that the zero set of f is neither a single point nor empty, then the bounding curve of the convex hull will be composed of a subset of $\mathcal{C}$ and a finite set of line segments whose endpoints are coincident with points on $\mathcal{C}$. In particular, each endpoint of these segments will either be tangent to $\mathcal{C}$ or a singular point of $\mathcal{C}$.

When computing the convex hull of a curved object, an interesting issue arises as to what form the output representation should take to ensure its usefulness for geometric calculations. In the case of point sets, the convex hull is a convex polygon which is readily represented by an ordered set of vertices. This representation is useful for many geometric tasks. For example, the boundary of the hull can be easily rendered by drawing the segments between successive points, and classifying a point $\mathbf{p} = (x,y)$ as being inside, on, or outside the hull is readily accomplished by considering the signs of the line equations defined by two successive points. In contrast, even rendering an algebraic curve is non-trivial because of the lack of a parametric form [3, 12, 22]; since the convex hull includes portions of the original curve, rendering these subsets is no easier. How to represent a subset of the curve is not obvious; for parametric curves, an interval of the domain can be easily used (see [9] for issues related to implicit curves). In this paper, a subset of an algebraic curve will be represented by a sample point $\mathbf{s} \in \mathbb{R}^2$ on the curve, and an oriented interval of the x axis; for rendering, points on the subset of $\mathcal{C}$ can be determined by tracing the curve from $\mathbf{s}$ to the interval bounds. This will be discussed more fully in section 2.4.3. Similarly, classifying a point $\mathbf{p}$ is also more difficult since the sign of $f(\mathbf{p})$ does not indicate if $\mathbf{p}$ is inside the curve (this will be considered more fully in section 2.3).

The rest of the paper is organized as follows: In section 2, an algorithm for constructing the hull is presented in detail including a sub-algorithm for point classification. The algorithm has been implemented, and this implementation and a few examples are presented in section 3. Finally, other issues and future directions are discussed in section 4.

2 Algorithm

The convex hull algorithm takes as input a curve $\mathcal{C}$ defined by an implicit algebraic equation in the form of (1). The output is a sequence whose elements are either straight line segments or subsets of $\mathcal{C}$ represented by a sample point and an interval of the x-axis. Figure 1 shows an algebraic curve with four components that we will consider throughout this discussion.

The algorithm is composed of the following steps:

1. Compute the set $\mathcal{L}$ of all line segments whose endpoints either are tangent to $\mathcal{C}$ or are singular points of $\mathcal{C}$.

2. Remove from $\mathcal{L}$ those segments that do not lie on the convex hull, yielding a set $\tilde{\mathcal{L}}$.

Convex Hulls of Algebraic Curves

David J. Kriegman and Erliang Yeh
Center for Systems Science
Department of Electrical Engineering
Yale University
New Haven, CT 06520-1968

Jean Ponce
Beckman Institute
Department of Computer Science
University of Illinois
Urbana, IL 61801

Abstract

A new algorithm based on curve tracing and decomposition techniques is presented for computing the convex hull of an algebraic curve defined implicitly by $f(x, y) = 0$; the curve may have multiple components as well as singular points. The output is an ordered collection of line segments and sections of the curve represented by a sample point and interval bounds; this representation is suitable for rendering the convex hull by classical curve tracing techniques. Additionally, we present a point classification function for the convex hull based on Sturm sequences. Progress toward extending these results to algebraic surfaces is briefly discussed.

1 Introduction

A classical problem in computational geometry is computing the convex hull of a set of points $\mathcal{P}$ in $\mathbb{R}^n$. Typically, $n = 2, 3$, and $\mathcal{P}$ is a collection of isolated points, line segments, curves or surfaces. The convex hull $\mathcal{H}$ is defined as the minimal convex set that includes all of $\mathcal{P}$, $\mathcal{P} \subseteq \mathcal{H} \subseteq \mathbb{R}^n$. In most work on this problem, $\mathcal{P}$ is a point set, and the hull $\mathcal{H}$ is a polytope which is readily represented as the intersection of a set of halfspaces. It has been shown that the convex hull of an arbitrary set of points in $\mathbb{R}^2$ can be computed in $O(n \log n)$ time [17].

Recently, there has been growing interest in the use of implicit algebraic curves and surfaces in geometric modelling and computer vision, however the host of computational algorithms developed for polygons and polyhedra is unavailable for these curve and surface representations. In particular, a number of researchers have been interested in extending convex hull construction algorithms beyond point sets, and have addressed the problem of computing the convex hull of curves in $\mathbb{R}^2$ and surfaces in $\mathbb{R}^3$. Schäffer and Van Wyk[19] consider piecewise smooth Jordan curves where each curve is represented parametrically. Similarly, Dobkin and Souvaine [7] have considered splinegons. More recently, Bajaj and Kim [4] have presented an algorithm for computing the convex hull of collections of algebraic curve segments defining a Jordan curve. Boissonnat et al. [5] have considered computing the convex hull of a set of n-spheres in $\mathbb{R}^n$.

In this paper, we consider the problem of computing the convex hull of an algebraic curve $\mathcal{C}$ in $\mathbb{R}^2$ defined as the zero set of an irreducible bivariate polynomial f of degree d:

$$f(x, y) = \sum_{i+j=0}^{d} c_{ij} x^i y^j . \tag{1}$$

Note that algebraic curves defined implicitly are more general than those defined parametrically (e.g. B-splines, NURBS, etc.) since every rational parametric curve has an implicit representation but the converse is not so [13, 21]. Also, implicit curves may include singular points (e.g. crossings, cusps, isolated points), may be composed of multiple components (e.g. an annulus), and may be unbounded (e.g. a hyperboloid). In this paper, we will assume that the curve is bounded (the hull of an unbounded curve such as a hyperboloid may be all of $\mathbb{R}^2$), but multiple components and singular points are permitted.