Accurate Computation of the Medial Axis of a Polyhedron

Tim Culver John Keyser Dinesh Manocha
Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175
{culver,keyser,dm}@cs.unc.edu

Abstract: We present an accurate and efficient algorithm to compute the internal Voronoi region and medial axis of a 3-D polyhedron. It uses exact arithmetic and representations for accurate computation of the medial axis. The sheets, seams, and junctions of the medial axis are represented as trimmed quadric surfaces, algebraic space curves, and algebraic numbers, respectively. The algorithm works by recursively finding neighboring junctions along the axis. It utilizes discretization of space and linear programming to speed up the search step. We also present a new algorithm for analysis of the topology of an algebraic plane curve, which is the core of our medial axis algorithm. To speed up the computation, we have designed specialized algorithms for fast computation on implicit geometric structures. These include lazy evaluation based on multivariate Stürm sequences, fast resultant computation, curve topology analysis, and floating-point filters. The algorithm has been implemented and we highlight its performance on a number of examples.

1 Introduction

The medial axis of a solid object provides useful shape information in terms of geometric proximity of its boundary elements. It is a skeletal representation that can be formulated as the locus of the center of a maximal sphere as it rolls around the object interior.

The medial axis of an object was first proposed by Blum [Blu67] for biological shape measurement. It has also been used for a number of other applications, including path planning, finite element mesh generation, automated injection molding simulation and feature recognition. Many researchers have proposed the medial axis transform as an alternate representation to B-rep and CSG in a design and interrogation system [Wol92, Hof94, SNTM92].

The medial axis of a solid is the closure of the locus of all points within the solid that have two or more closest points on the boundary of the solid. When the solid is a polyhedron, the medial axis is composed of bisectors of boundary features (vertices, edges, and faces). These bisectors are planes and quadric surfaces. The medial axis consists of *sheets*, *seams*, and *junctions*. A sheet is the bisector of two boundary elements, and may be represented as a trimmed quadric surface. A seam is an algebraic space curve defined by the intersection of two or more sheets. A junction point is defined by the intersection of three or more sheets.

The complexity of the medial axis is not fully understood. The total number of sheets, seams, and junctions can be $O(n^2)$, where n is the total number of faces, edges, and vertices in the polyhedron; it may even be $O(n^3)$. Many polyhedral medial axis algorithms use either a spatial subdivision or some form of discrete representation of the surface. Sheehy et al. [SAR95] claim that because of the inherent complexity of a continuous approach, one has to use some form of discretization. However, it is hard to accurately and efficiently compute the medial axis based on such a discrete representation. In the last few years, a number of authors have proposed computing the medial axis based on a tracing approach [Mil93, Chi92, SPB95]. Starting with a known junction (for instance, a vertex of the polyhedron), they trace along its incident seams and discover the adjacent junction points. In particular, Sherbrooke et al. [SPB95] have presented the first implemented algorithm for computing the medial axis based on a continuous representation of the boundary. However, they compute numerical solutions of polynomial equations using floating point arithmetic, and compute piecewise linear approximation of the seam curves. The accuracy varies with the step size and the tolerances used for computing the junction points. Since the medial axis of a polyhedron can be very sensitive to perturbations, it can be rather non-trivial to design an accurate algorithm using finite precision arithmetic. For example, figure 13 shows the medial axis of a polyhedron which has four junction points very close to each other. In practice, it is difficult to accurately compute the medial axis of such a polyhedron using finite precision arithmetic.

The problem of accurate and robust computation using finite precision arithmetic is a major open problem in solid modeling [Hof89]. One solution to the precision problem is to use *exact arithmetic*. For example, many algorithms based on exact arithmetic have been proposed for reliable computation for boundary evaluation of boolean combinations of polyhedral models [SI89, For95, BMP94, Hof89]. However, medial axis computation involves low-degree nonlinear algebraic curves and surfaces. For such problems, it is difficult to compute tight bounds on the numerical error generated due to floating-point arithmetic. As a result, it is hard to design reliable algorithms using tolerance-based approaches. At the same time, progress is slow in the application of exact arithmetic to nonlinear problems, mainly because exact arithmetic is perceived to be impractically slow.

In this paper, we present an accurate and efficient algorithm to compute the internal Voronoi region and medial axis of a 3-D polyhedron. Our overall approach is similar to the 3-D tracing algorithm proposed by [Mil93, Chi92, SPB95]. However, we use exact arithmetic and representation for accurate computation of the medial axis. The algorithm recursively computes the vertices and edges of the medial axis based on seam-tracing. We also present a new curve topology evaluation algorithm for reliable tracing of the seams of the medial axis. Our contributions include:

- **Search algorithm:** An accurate and reliable algorithm for evaluating all the seams and computing the "first" junction point lying on the seam.
- **Representation:** Efficient and exact representations of junction points, seams

and sheets of the medial axis.

- **Efficiency:** We use techniques based on space discretization and linear programming to speed-up the search step and avoid enumerating all possible combinations of junction points. To speed up arithmetic operations on algebraic numbers, we have designed specialized algorithms for fast evaluation and computation of the junction points, seams and sheets. These include lazy evaluation based on multivariate Stürm sequences, fast resultant computations, curve topology evaluation, and floating-point filters.
- **Handling Degeneracies:** We identify the cases where degeneracies can affect our algorithm, and propose ways to identify and resolve them.

The resulting algorithm has been implemented and we highlight its performance on some example polyhedra. Its overall complexity is output sensitive and depends on the combinatorial complexity of the medial axis.

Organization: The rest of the paper is organized in the following manner. We survey the previous work in medial axis computation and exact arithmetic in section 2. Section 3 gives an overview of our algorithm and describes the exact representations of the elements of the medial axis. In section 4, we describe our new search and curve topology analysis algorithms. We present a number of techniques to improve the efficiency of our approach in section 5. Section 6 discusses degeneracies in the medial axis computation and how we handle them. We describe our implementation in Section 7 and highlight its performance on a few polyhedra.

2 Previous work

2.1 Medial axis computation

There is considerable work on medial axis computation in solid modeling and computational geometry. The various approaches can be classified into two main categories: discrete methods based on sampling, and "continuous" or direct methods that do not rely on the sufficiency of some sampling.

Sampling points on the surface: Many authors have computed the medial axis using the Voronoi diagram of a set of points located on the polyhedron's surface. Usually the result is an approximation to the medial axis, though Sheehy et. al. [SAR95] adaptively refine the sampling until all adjacency relationships are revealed. From this, the exact medial axis may be constructed. However, the overall complexity of this approach is not understood.

Spatial Subdivision: Another popular approach is to impose a regular grid on space, and compute an approximation to the medial axis consisting of a set of cells. The approximation can be considered an "image" of the medial axis, and does not contain a complete description of the adjacency relationships. Among implementations of this approach, that of Vleugels and Overmars [VO95] is notable in that they guarantee correct topology of the approximate axis. Their topology-resolution routine subdivides adaptively. Thus, like the boundary-discretization method, the complexity is difficult to formulate in terms of the input size.

2-D Continuous Algorithms: The classic divide-and-conquer algorithm for the two-dimensional problem has been presented by Lee [Lee82]. A variant of that algorithm is given in [Hel97]. Fortune has presented a sweepline algorithm [For87]. No algorithms are known in the literature that extend these approaches to 3-D. Imai has presented an incremental algorithm for 2-D polygons [Ima96].

Incremental 3-D Algorithm: Milenkovic has presented an an incremental algorithm for computing the voronoi region of a polyhedron. It inserts the boundary elements one at a time. At step k, the entire generalized Voronoi diagram of the first k sites has been constructed. However, we are not aware of any implementation of this algorithm.

3-D Tracing Algorithms: All the practical algorithms for computing the 3-D medial axis are based on the *tracing* approach. Starting from a junction point, a seam emanating from the junction is followed. The seam terminates at another junction. Once this point is found, the algorithm recursively forks and follows all seams emanating from that junction. The key step is the search for the seam terminator. In the two-dimensional medial axis problem, the search is limited to smaller and smaller subchains of the polygon. No such property holds for the three-dimensional problem; each terminating junction is found at the expense of a search of (in the worst case) the entire polyhedron. Thus the complexity is jointly proportional to the input size n of the polyhedron and the output size m of the medial axis. The algorithm's worst-case behavior is necessarily at least a factor of n larger than optimal $O(m)$ time. A tight upper bound on m, the size of the medial axis, is currently not known. It is known that $m = O(n^{3+\epsilon})$ for any $\epsilon > 0$ [AAS97], but we know of no example where m is more than $O(n^2)$.

Milenkovic [Mil93] proposed the 3-D tracing algorithm, and discusses its complexity. Chiang [Chi92] presented an algorithm for computing the medial axis of a planar region bounded by piecewise C^2 curves. It involves tracing branches using sets of polynomial equations. Sherbrooke, Patrikalakis and Brisson [SPB95] have presented a variation on the algorithm. They explicitly trace along the seam, creating a piecewise-linear approximation to the curve. They applied their algorithm to generate the medial axis of a number of polyhedra. Reddy and Turkiyyah [RT95] also present a version of the tracing algorithm, and compare it to an algorithm based on the Voronoi diagram of a set of points on the boundary.

Medial Axis of CSG Objects: Dutta and Hoffmann [DH90, DH93] and Hoffmann [Hof94] have presented algorithms for computing the medial axis of CSG objects. Their algorithms compute the points of closest approach between pairs of boundary elements. They have also studied exact representation of bisectors arising in medial axis computation of CSG objects bounded by planes, quadrics and torii.

Medial axis to B-rep Conversion: A number of algorithms have been presented in the literature to convert solids from a representation based on the medial axis to a boundary representation [Wol92, Chi92, Bra92, Ver94].

2.2 Exact arithmetic in geometric computation

Exact arithmetic has proven useful in the linear domain. Exact arithmetic means that each number is determined and stored to whatever precision is necessary. By allowing arbitrary bit-lengths any integer or rational number can be computed and stored exactly. For linear objects, rational numbers are usually all that is required to perform a geometric operation. In the solid modeling community, exact arithmetic has been used successfully for applications involving linear objects (e.g. [For95]). Some of the recently developed geometric libraries like LEDA and CGAL also provide support for exact arithmetic.

In the nonlinear domain, however, exact arithmetic becomes more difficult to apply effectively. This is because computations on non-linear objects often require algebraic numbers, which cannot be explicitly represented by a finite number of bits. Techniques using bit-length estimates may, in the worst case, require bit-lengths which are exponential with respect to the degree of the algebraic functions [Can88, Yu92]. One approach to this problem is to use field extensions to allow representations of each new number as it is computed. Many computer algebra systems and symbolic libraries provide support for field extensions. However, computing with such numbers can be quite expensive. Another approach is to represent an algebraic number as the unique root of a polynomial (with rational coefficients) within a rational interval. This is similar to the idea of interval arithmetic. As long as the interval containing the root can be tightened or cut on demand, many useful queries can be performed exactly using this representation. In one dimension, this can be accomplished through the use of Stürm sequences. In higher dimensions, *multivariate Stürm sequences* are used. Milne [Mil92] and Pedersen [Ped91] have proposed methods for such computations. Keyser et. al. apply these methods to the problem of CSG boundary evaluation [KKM97]. The authors demonstrate that the geometric operations for this problem can be computed in rational surfaces using only 1-D and 2-D Stürm sequences. The medial axis problem requires the use of either 3-D Stürm sequences or field extensions.

3 Algorithm Overview

Our approach builds on the tracing algorithm proposed by [Mil93], [SPB95], and [RT95]. However, instead of computing a piecewise linear approximation, we exactly compute all the components of the medial axis. Essentially, the algorithm constructs the adjacency graph of the medial axis while traversing it. Our terminology is similar to that of [SPB95].

The input polyhedron consists of *faces*, *edges*, and *vertices*. The coordinates of the vertices are represented in terms of rational numbers. The faces, edges, and vertices are collectively called *boundary elements*. The medial axis consists of *sheets*, *seams*, and *junctions*. The sheets correspond to trimmed quadric surfaces. The seams are algebraic curves with rational coefficients and the junctions are points whose coordinates correspond to algebraic numbers (of degree at most 8). A sheet is said to be *governed* by two boundary elements, a seam by three (or more), and a junction by four (or more). We denote the *squared distance function* associated with a point, line, or plane by $d_e^2(x, y, z)$, where e stands for the point, line, or plane. The squared distance function is a quadratic formula in x, y, and z.

The algorithm starts by finding a single seam. For example, if there is a vertex of the polyhedron with three incident edges, all of which are convex, then its three incident faces generate a seam. If no vertex of the polyhedron is trivalent, a more sophisticated technique is used to get the algorithm off the ground with an initial seam. This is discussed in section 6.1.

Each remaining boundary element e in the polyhedron is then considered as a potential quadruple (e_1, e_2, e_3, e) generating a junction point on the seam. One seeks the element e_4 whose candidate junction $\mathbf{p}_1$ comes first along the seam, measured by arc length. The ingredients of the search are

- the seam governors e_1, e_2, e_3;
- the starting point $\mathbf{p}_0$, a junction point governed by, say, (e_0, e_1, e_2, e_3);
- the search direction $\mathbf{w}$, one of the two tangent directions at $\mathbf{p}_0$.

The determination of e_4 and $\mathbf{p}_1$ is based, not on taking tiny steps along the seam, but on an accurate topological analysis of the curve.

The newly-found junction point may be a vertex of the polyhedron. In this case, the branch of computation finishes, and an unsearched seam is taken from a priority queue. Otherwise, the junction point is equidistant from (ignoring degeneracy for now) four boundary elements (e_1, e_2, e_3, e_4). Assuming the junction point has not already been visited, three new seams are inserted in the priority queue: those corresponding to (e_2, e_3, e_4), (e_1, e_3, e_4), and (e_1, e_2, e_4). When the priority queue is empty, the algorithm terminates.

For each new seam, the search direction is decided by comparing the curve tangent to the three bisector surfaces involving the discarded element. The correct tangent direction lies on the non-discarded side of each of these three surfaces. The *medial axis transform* of an object consists of is medial axis together with a *radius function*, which assigns to each point on the axis the radius of the maximal ball at that point. The radius function is constructed by composing the parametric form of a sheet, quadratic in s and t, with the distance function to either of its governors, a quadratic in x, y, and z. The radius function is in general a degree-four expression in s and t.

3.1 Exact Arithmetic

We assume that the boundary of the polyhedron can be represented using rational numbers. However, there is a fundamental limitation to the use of exact rational arithmetic in Euclidean geometry: constructions on rational numbers often result in algebraic numbers. For an example in the plane, consider the set of points equidistant from the line $y = 0$ and the line $y = x$: the "bisector" of these two lines is the union of the lines $y = (-1 + \sqrt{2})x$ and $y = (-1 - \sqrt{2})x$. Evidently, these lines cannot be described using only rational numbers. Their equations have algebraic numbers of degree 2, and writing them in "two-point form" does not fix the problem, since the origin is the only point on either line with rational x and y coordinates.

Exact arithmetic with these numbers is possible by computing in an extended field. In the example above, this amounts to representing all numbers as $a + b\sqrt{3}$, where a and b are rational. However, computing in an extended field is significantly more expensive than computing over the rationals, especially when the field extension grows over the computation.

We have chosen to avoid arithmetic involving such algebraic numbers, choosing instead to design implicit geometric structures for those objects which cannot be described with rational numbers, and using only rational arithmetic. In the example above, we would represent the pair of lines together as the degenerate conic $-x^2 + 2xy + y^2 = 0$. The choice not to separate the two lines (for usually only one is relevant) will complicate our algorithms somewhat. The expense of dealing with a degree-2 representation for a degree-1 object is analogous to the expense of dealing with the quadratic field extension.

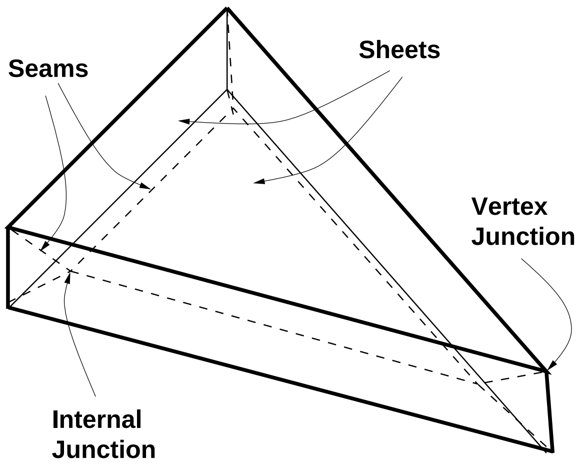

Figure 1: *The geometric elements of the medial axis*

3.2 Exact geometric representations

This section describes our exact representations for the sheets, seams, and junctions of the medial axis. The parts are illustrated in figure 1.

A sheet is the bisector of two boundary elements. These are quadric surfaces, amenable to two different representations: an implicit form

$$F(x,y,z)=0$$

where F is a polynomial of total degree 2, and a rational parametric form

$$\left(\frac{X(s,t)}{W(s,t)},\frac{Y(s,t)}{W(s,t)},\frac{Z(s,t)}{W(s,t)}\right)$$

where X, Y, Z, and W each are polynomials of total degree 2. The implicit form is just the difference of two distance functions, and has rational coefficients. The parametric form, however, may have irrational coefficients, depending on the two governors and their configuration. Our medial axis algorithm always uses the implicit form, and also uses the parametric form whenever its coefficients are rational.

The parametric form is computed directly from the governors. For instance, the bisector of two skew lines can be parametrized by letting s run along one line and t along the other. The point at the intersection of the normal plane at s, the normal plane at t, and the bisector plane of s and t lies on the bisector surface. When s and t are rational, the point on the bisector is rational. This plane-intersection technique, presented for rational curves in space in [EK96], works for other bisector types as well. It cannot work, however, for the generic line-plane and plane-plane bisectors, since they are largely devoid of rational points.

A seam curve lies on three different sheets; it is represented as the intersection of any two. An explicit rational parameterization of these curves is, in general, impossible; thus they are represented in implicit form only. The implicit form $G(s,t)=0$ is obtained by substituting the parametric form of one surface into the implicit form of

Governors	*Configuration*	*Bisector*	*Coeffs in Q?*
Point-point	Generic	Plane	Yes
Point-line	Generic	Parabolic cylinder ($z = x^2$)	Yes
	Incident	Plane	Yes
Point-plane	Generic	Paraboloid ($z = x^2 + y^2$)	Yes
	Incident	Line	Yes
Line-line	Generic	Saddle surface ($z = x^2 - y^2$)	Yes
	Incident	Orthogonal plane pair ($z^2 + x^2 = 0$)	No
	Parallel	Plane	Yes
Line-plane	Generic	Right circular cone ($z^2 = x^2 + y^2$)	No
	Perpendicular	Right circular cone	Yes
	Parallel	Parabolic cylinder	Yes
	Line lies on plane	Plane	Yes
Plane-plane	Generic	Orthogonal plane pair	No
	Parallel	Plane	Yes
	Perpendicular	Orthogonal plane pair	Yes

Figure 2: *The surfaces that arise as sheets of the medial axis.*

the other. If none of the surfaces has a rational parametric form with rational coefficients, an implicit form is obtained by projecting the curve onto a plane (see section 4.2.3). Computations involving the curve are usually formulated in the (s, t) plane.

A junction is an *algebraic point*, represented jointly by

- a system of three algebraic equations in three variables with rational coefficients;
- a three-dimensional box with rational coordinates containing exactly one root of the system.

An implementation of trivariate Stürm sequences in exact arithmetic can find a box with a single root, and refine its size arbitrarily. Reasoning about the point typically involves reasoning about the box, while shrinking the box until a query can be answered unambiguously. A vertex-junction is equidistant from all of its incident edges and faces, and its coordinates are known exactly, so there is no need to associate any of the above structure with such a junction.

4 The searching algorithm

In this section, we enumerate all possible morphologies for bisector surfaces and seam curves based on all combinations of governors. Based on this classification, three distinct seam-search algorithms are presented. Each is suited to a different configuration, and they may be thought of roughly as evaluation of curves in one-dimensional, a two-dimensional, and a three-dimensional space.

4.1 Bisector morphology

In figure 2, we show all of the bisectors that arise in the medial axis of a polyhedron. Note that some of the non-generic configurations arise in perfectly generic polyhedra. For instance, the bisector of a plane and a line in the plane occurs between the Voronoi

Governors	*Generic seam*	*Analysis method*
Point-point-point	Line	4.2.1
Point-point-line	Parabola	4.2.1
Point-point-plane	Conic	4.2.1
Point-line-line	Quartic	4.2.2
Point-line-plane	Quartic	4.2.2
Point-plane-plane	Conic	4.2.2
Line-line-line	Quartic	4.2.2
Line-line-plane	Quartic	4.2.2
Line-plane-plane	Conic	4.2.3
Plane-plane-plane	Line	4.2.3

Figure 3: *Some curves that arise as seams in the medial axis.*

region of a face and that of an adjacent (non-convex) edge. The last column refers to the coefficients of the parametric form of the surface; the implicit form always has rational coefficients. The one bisector that is not a surface—that of a plane and a point on the plane—can be safely ignored in our medial axis algorithm.

Our medial axis algorithm uses exact representations of the bisector surfaces. An implicit form with rational coefficients is always available, but in the indicated cases, the parametric form may involve square roots. The consequences of this for our algorithm are discussed in section 4.2. In the output, these surfaces can be given in parametric form with algebraic coefficients of the form $a + b\sqrt{c}$ for rational a, b, c, or they can be approximated by parametric surfaces of the same geometric type with rational or floating-point coefficients with bounded error.

4.2 Seam morphology

The ten combinations of boundary elements are listed in figure 3, with the most general seam curve type for each combination. The "analysis method" corresponds to a choice of three algorithms, described in the subsubsections below. The degenerate cases are too numerous to list here.

4.2.1 Rational Parametric Curves

Sorting points along a rational parametric curve is a one-dimensional problem. Each candidate junction is represented as an algebraic number, in implicit form (a polynomial and a rational interval containing exactly one root).

The coefficients of this parametric form may not be rational. In these cases, it is feasible to perform the computation in an extension field of the rationals, but we find it more efficient to treat the curve as if it had no parametric form, as this avoids the computational expense of handling field extensions.

4.2.2 Non-rational Curves lying on Rational Surfaces

In many cases, we are unable compute a rational parametrization of the curve with rational coordinates. However, we may have a parametrization in rational numbers of a surface that contains the curve. In such cases, we form an implicit representation of the algebraic curve in the parametric domain of the surface.

In the cases represented by the first eight rows of 3, at least one of the three bisector surfaces admits a rational parametrization with rational coefficients. We use that parametric representation to evaluate the seam curve. Using the parametric form of the surface, we project the curve back to the (s,t) plane. The topological analysis algorithm (described in section 4.4) decomposes the curve into monotonic segments. For each candidate boundary element, we choose the bisector surface associated with one of the seam governors. The intersection curve with S is pulled back to the same (s,t) domain. Intersections of the seam curve with the candidate-curves in (s,t)-space correspond to potential junction points. Each potential junction point is located in one of the monotonic boxes. It is then a simple matter to order the candidate junctions and find the first one along the seam curve.

4.2.3 Non-rational Curves lying on Algebraic Surfaces

In the last two seam combinations, line-plane-plane and plane-plane-plane, the bisector surfaces (cones and plane-pairs) may not have rational parametric forms with rational coefficients. Rather than using arithmetic in an extension field, we note that in these two cases (including degenerate configurations) the seam curve is always a conic or a line, and elect to treat these cases with specialized three-dimensional methods. The candidate junctions are all isolated in (x,y,z)-space using tri-dimensional Stürm sequences.

For the generic plane-plane-plane case, the equidistant point set corresponds to four lines meeting at the intersection of the planes. The various slopes of these lines are all, in general, irrational. The seam is contained in one of the eight "branches" of this curve, and the plane equations of the three governors distinguish points on this branch from points on the others. The ordering among the candidate junctions is determined by, say, their x-coordinates.

For the generic line-plane-plane case, the curve consists of two non-intersecting conics lying in perpendicular planes; the planes' various slopes are generally irrational. Points on the irrelevant conic can be culled away by testing against the plane equations of the two governing planes. The curve, represented implicitly by the intersection of two bisector surfaces $f(x,y,z)=0$, $g(x,y,z)=0$, is then projected onto the xy-plane by eliminating z from the pair of surface equations using the standard Sylvester resultant. Before projecting, we transform the system so that eliminating z corresponds to projection onto a plane close to the plane in which the conic lies. The projected curve is a quartic in x and y. It is the product of two conics, but cannot be factored over the rationals. A suitable domain is chosen and the curve topology routine is run on the quartic curve. The boxes containing the candidate junctions project to hexagons. If the projections overlap, the points are shrunk in 3-D and reprojected. Figure 4 shows an example where the seam is an ellipse lying in an irrational plane. Not shown, and not part of the seam, is a hyperbola in a plane perpendicular to the ellipse's plane; under projection to a rational plane, the hyperbola will appear.

4.3 Local geometry of Junction Points

This section addresses the issue of discovering the seams that are incident to a junction. Consider a junction governed by elements $\{e_1,\ldots,e_k\}$ where $k \geq 4$. In the simplest case, $k=4$ and each of the three-element subsets of $\{e_1,e_2,e_3,e_4\}$ govern a seam. But

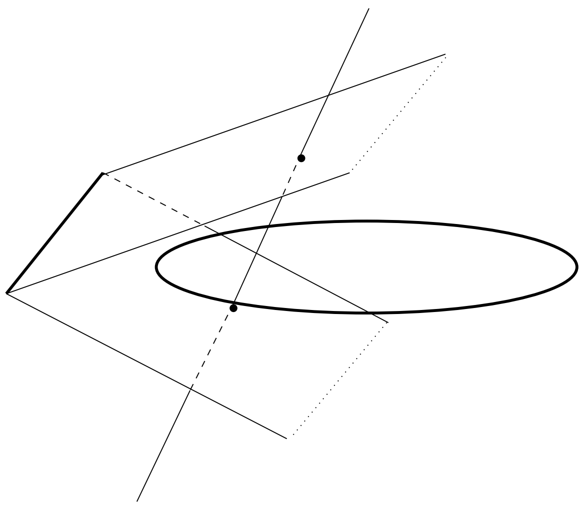

Figure 4: *An elliptical seam in a plane with irrational slope*

each of these seams can be followed in either of two directions. And when $k > 4$, not all three-element subsets actually govern seams. These decisions are made by comparing seam tangent vectors with bisector tangent planes.

Junction points can be seen to fall into two categories: *internal junctions* and *vertex-junctions*. An internal junction $\mathbf{q}$ is characterized by:

- $\mathbf{q}$ is equidistant from four or more boundary elements;
- this distance is positive, and q lies in the interior of the polyhedron;
- $\mathbf{q}$ being equidistant from five or more boundary elements is indicative of a truly degenerate situation;
- the coordinates of $\mathbf{q}$ are algebraic numbers of degree at most eight.

A vertex-junction $\mathbf{v}$ is characterized by:

- $\mathbf{v}$ coincides with a vertex of the polyhedron;
- $\mathbf{v}$ is equidistant from several boundary elements, but this distance is zero;
- the number of equidistant boundary elements is quite commonly more than four, as it includes all incident faces and (reflex) edges;
- the coordinates of $\mathbf{v}$ are rational by assumption.

It is clearly advantageous to distinguish these two types of junction point in the implementation. It can be argued (see [SPB95]) that the connectivity properties of the medial axis make it feasible to avoid searching out of vertex-junctions at all—they serve as "sinks" for the recursion. Doing so typically removes many degenerate junctions from consideration. However, we still need to deal with the problem of making correct decisions at degenerate interior junctions.

For any type of junction—interior, vertex, degenerate, non-degenerate—the following predicate decides whether a potential seam actually participates in the medial axis. Let E be a subset of the subset of junction governors, and $\tilde{E}$ be its complement, so that $E \cup \tilde{E} = \{e_1, \ldots, e_k\}$. Of course E must have at least three elements. For each pair (e, e') where $e \in E$ and $e' \in \tilde{E}$, let $h(e, e')$ be the tangent plane to the bisector surface of e and e' at the junction point. Let $\mathbf{w}$ be one of the two tangent vectors to the curve equidistant from the elements in E. The seam defined by E and $\mathbf{w}$ participates in the medial axis if and only if for every pair (e, e'), the vector $\mathbf{w}$, compared to $h(e, e')$, points toward e and away from e'. Note that a particular E may not generate a seam, and that if $(E, \mathbf{w})$ does generate a seam, then $(E, -\mathbf{w})$ cannot.

There are $\binom{k}{3}$ possible seams incident to our vertex, ignoring the possibility that a seam may be governed by more than three elements. Checking a seam requires constructing $3(k-3)$ tangent planes. Checking all possible seams, then, requires $\frac{1}{2}k^4 - 3k^3 + \frac{11}{2}k^2 - 3k$ vector-versus-plane checks. For a non-degenerate interior vertex, we know there is a seam for each E, and it is just a matter of choosing which $\mathbf{w}$ goes with each one; this requires only 4 tangent plane checks. For a very degenerate vertex, though, the exhaustive method is too expensive. The combinatorial structure of the seams incident to a junction point has the structure of a cell decomposition of the surface of a sphere, and perhaps can be characterized as some sort of Voronoi diagram, and computed efficiently as such. This is an interesting issue for future research.

Comparison of the vector $\mathbf{w}$ to the plane h is performed in exact arithmetic as follows. The seam is represented implicitly in $\Re^3$ as the intersection of two surfaces. A tangent $\mathbf{w}$ may be found by crossing the gradients of the two surfaces. The plane h is the orthogonal complement of the gradient $\mathbf{v}$ of the bisector surface $d^2_{e'}(x, y, z) - d^2_e(x, y, z) = 0$. When the surface is written this way (instead of in the opposite order), the gradient points "towards" e, so the predicate on $\mathbf{w}$ can be evaluated as the sign of $\mathbf{w} \cdot \mathbf{v}$. The latter expression is the determinant of the three gradient vectors. The determinant is expanded symbolically in terms of x, y, and z, and forms a cubic polynomial in those variables. If the rational box containing the junction point lies entirely on one side of the zero set of this cubic polynomial, then the sign can be evaluated at any of the eight corners.

4.4 Curve Topology Algorithm

In this section, we present a specialized curve topology algorithm to evaluate the seam curves. The algorithm divides an algebraic plane curve into segments which are monotonic in both s and t. Such a division allows points on the curve to be easily ordered along it. Although general algebraic decomposition algorithms (e.g. [AF88]) can be used on this problem (after being modified to handle the "root-in-a-box" format), we have developed our own algorithm which is more specialized to ordering points along the seam curve.

We will present only a high-level description of the algorithm here. An example of the output of our algorithm is shown in figure 5.

The algorithm we use makes two key assumptions. First, it assumes that the algebraic plane curve contains no self-intersections or other singularities in the region of interest. Although this significantly restricts the curves we can use the algorithm on, the curves which arise in the medial axis computation generally meet this requirement. Second, we assume that all points on the curve are computed distinctly—that is, their containing rational boxes do not overlap.

First, compute all of the turning points of our algebraic plane curve, $f(s, t) = 0$. This is done by finding all common solutions between $f = 0$ and $f_s = 0$, and between

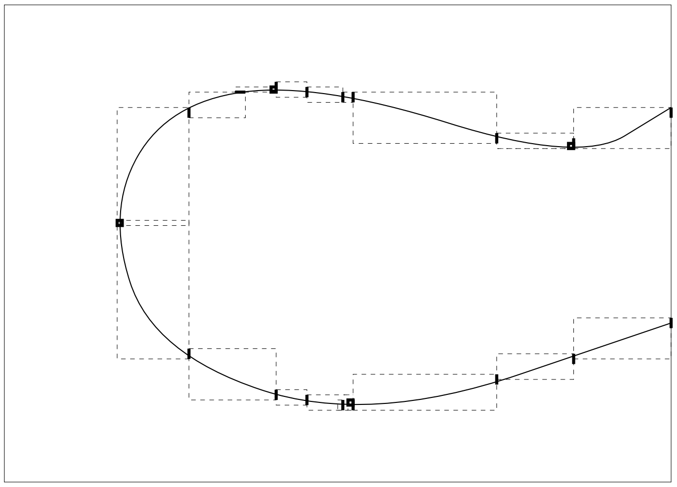

Figure 5: *The output of the curve topology algorithm.*

$f = 0$ and $f_t = 0$, subscripts denoting partial derivatives. This will isolate all of the turning points in s and t, as well as inflection points. Next, we find all of the "edge points," intersections of $f = 0$ with the boundaries of the region of interest (which is assumed to be rectangular and axis-aligned).

The next step is to determine the connections between the turning and edge points. The curve between any two of these connected points will be monotonic in both s and t. To make these connections, the algorithm proceeds by recursively subdividing rectangular regions. A region falls into one of the following categories:

- Contains more than one turning point
- Contains one turning point and more than two edge points
- Contains one turning point and exactly two edge points
- Contains no turning points

Treating each of these cases separately, we recurse until all regions fall into the last two categories.

At this point, the curve $f = 0$ has been divided into a number of segments monotonic in both s and t in the region of interest. Finally, we subdivide further until the bounding boxes of the curve segments do not overlap.

5 Improving the Efficiency

Our implementation uses a lower level library to perform exact computations with algebraic numbers. We refer to them as *kernel operations*. At the top level, the algorithm spends most of its time in the search step, typically involving one curve topology invocation and the isolation of many candidate junction points in 2-D or 3-D.

In this section, we present three sets of techniques that improve the efficiency of the overall algorithm. These are:

- Speeding up the search algorithm using discretization of space and linear programming.
- Reducing the number of kernel operations and root computations.
- Fast evaluation of kernel operations.

5.1 Speeding up the Search Algorithm

For 3-D Stürm computations, the shrinking of a box around a root is cheap compared to the expense of setting up the system. The efficiency of the searching step is significantly affected by the number of candidate elements checked against the seam analysis. Any inexpensive method for rejecting candidate boundary elements without examining the potential junction points is likely to speed up the algorithm.

We illustrate two methods for reducing the number of boundary elements considered during the searching step. Neither idea reduces the asymptotic worst-case complexity of the algorithm. Both are essentially culling methods, capable of choosing some elements which cannot govern certain junctions.

5.1.1 Discretization of space

Milenkovic [Mil93] has proposed the following method for speeding up the search step. As a preprocess, subdivide space recursively, yielding an octree of axis-aligned boxes. A simply-computed predicate, applied to each box, yields a (hopefully small) superset of the set of boundary elements whose Voronoi regions intersect the box.

The searching step of the medial axis algorithm then proceeds as follows, in the cases where the 2-D curve topology step (section 4.2.2) is required. Similar strategies may be invented for the other cases (sections 4.2.1 and 4.2.3). The first 2-D box in the topology diagram is considered as the domain of a Bézier surface forming part of the bisector surface S. The control points of this Bézier surface are easily computed from the parametrization of S, and the 3-D bounding box of the control points contains the relevant segment of the seam curve. The algorithm then descends the octree, finding a minimal collection of nodes that cover the seam's bounding box. If the desired junction point lies on this first curve segment, the boundary element that governs it will be among the union of all of the closest-element-supersets of the nodes. Otherwise, the algorithm moves to the second monotonic section of the seam curve in the domain of S and repeats.

This strategy requires a heuristic for the termination of the preprocess. Milenkovic [Mil93] essentially recommends letting it run (in breadth-first order, finishing a level entirely before subdividing again) until the running time becomes significant compared to the time which will be spent in computing the medial axis.

5.1.2 Linear programming

An alternate pre-process for speeding up the search algorithm is based on linear programming. For each active boundary element, the preprocess finds a polytope which is known to contain the Voronoi region of the element. Only the structure local to the element is examined. Some examples are given in figure 6. The polytope can be made finite by drawing a bounding box around the element and taking half the

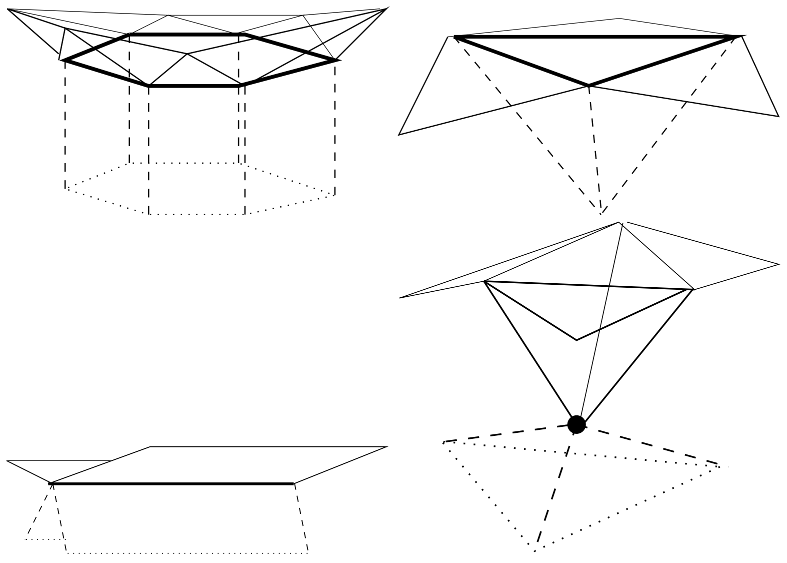

Figure 6: *Polytopes containing the Voronoi regions of (a) a face with reflex edges, (b) a face with convex edges, (c) a reflex edge, (d) a reflex vertex.*

distance from each of these six planes to the corresponding plane of the bounding box of the whole polyhedron. The number of planes in such a polytope is proportional to the number of edges around a face or the number of edges incident to a vertex—for practical purposes, a constant.

Next, the algorithm takes each pair of active boundary elements and tests their polytopes for mutual intersection. With each element is associated a sorted list of the other elements who are potentially Voronoi neighbors. When the seam equidistant from elements e_1, e_2, e_3 is traced, the desired element e_4 must be a Voronoi neighbor of all three of the seam generators. The algorithm then limits its attention to the elements which are on all three lists.

The preprocessing involves $\binom{n}{2}$ steps, each of which performs a linear programming test (constant time) and a list insertion ($O(\log n)$ time). The overall preprocessing running time is then $O(n^2 \log n)$. The asymptotic running time of the medial axis algorithm is not increased, since the list-intersection step takes at worst $O(n)$ time for each seam traced—the same amount of time that was already committed to examining all the other boundary elements.

Thus the preprocessing time of $O(n^2 \log n)$ is asymptotically small compared to the worst-case complexity of the medial axis algorithm, $O(n^4)$, and not much greater than the best-case medial axis complexity $O(n^2)$. Moreover, the constants in the preprocessing time are much smaller than those in the medial axis implementation, since the linear programming step can be programmed in floating-point. Fast randomized algorithms for linear programming are well known [Sei90].

5.2 Reducing the number of kernel operations

Roots of systems are only computed to the precision necessary. Our representation is exact, but implicit, and the geometric algorithms usually pay attention only to the rational box containing the root. At all times, the rational box is shrunk only as much as necessary to answer the query. This is a form of "lazy evaluation" of solutions to polynomial systems.

The efficiency of floating-point arithmetic can be leveraged against the problem of finding roots to systems. While refining roots is based on bisection, an initial estimate to the root can be based on a floating-point approximation. One Stürm test is executed to verify the correctness of the approximation to within a certain precision. In cases where the floating-point solver produces a correct answer to within the desired precision, no further Stürm operations are needed.

Sometimes a root of a system of equations may be isolated by projecting the system onto a lower-dimensional space. Given two bivariate polynomials, $f(s,t)$ and $g(s,t)$, we can eliminate (using a Sylvester resultant) one of the two variables, leaving us with a univariate polynomial, say $F(s)$. The roots of $F = 0$ correspond to the s coordinates of the common solutions of $f = 0$ and $g = 0$. We can examine these projections of each root and count the number of roots contained in the projections by using univariate Stürm calculations. If there is only one root in the interval for *both* the s- and t-projections, then we can use the two univariate formulations instead of the multivariate formulation in order to refine the root. Although the switch to a univariate formulation may not be possible when the root is known to only a rough precision, as the precision is refined, it becomes more likely that a univariate formulation will suffice, since as the box bounding the root is tightened, the projections of that box become smaller and less likely to contain the projections of other roots.

5.3 Speeding up the kernel operations

Our implementation spends most of its time in the Stürm sequence code, refining roots of systems in two and three dimensions. A multivariate Stürm sequence computation is executed in two main stages. In the *elimination* stage, a system of many equations in many unknowns is reduced to a single polynomial, the volume function. The *sequence evaluation* stage counts the number of sign changes in the Stürm sequence of the volume function.

5.3.1 Elimination

The elimination stage essentially turns a multivariate root-counting problem into a univariate root-counting problem. It involves (in the trivariate case) adjoining an extra equation $u + (\alpha - x)(\beta - y)(\gamma - z)$, substituting in various combinations of rational numbers for α, β, and γ corresponding to the corners of the box, and then eliminating x, y, and z from the resulting system of four equations.

For two-dimensional systems, the elimination stage is accomplished by means of three Sylvester resultants. The repeated resultants introduce extra factors into the volume function, but these are easily identified and quickly removed. The elimination process is quick, and the time is spent mainly in the sequence evaluation stage.

For three-dimensional systems, repeated use of Sylvester resultants can result in extraneous factors. Instead, we use Macaulay's resultant [Mac02], which gives the volume function as the quotient of the determinant of a matrix M and the determinant of a sub-matrix of M. When the system consists of three quadrics in x, y, and z, as it does in the worst case for the medial axis problem, M is an 84×84 matrix whose entries are constants or linear polynomials in the variable u. We utilize the sparse representation of the matrix to quickly evaluate the determinants.

The system can be arranged so that the Macaulay denominator depends only on the coefficients of the system, and does not contain α, β, γ, or u. The determinant is then a constant, so no polynomial division is necessary. One only needs to check the denominator matrix for singularity. The denominator is also independent of the point (α, β, γ) in space at which the Stürm sequence is evaluated, so it need be checked only once per system.

The Macaulay numerator is arranged into block form:

$$\begin{pmatrix} A & B \\ C(\alpha,\beta,\gamma) & D(\alpha,\beta,\gamma) + uI \end{pmatrix}$$

where A and B contain coefficients of the system, and C and D depend on the point (α, β, γ). A is 76×76, while D is 8×8. By performing block Gaussian elimination on this matrix, the bulk of the work of taking the determinant of the Macaulay numerator is performed before α, β, γ are known. The precomputation consists of finding the LU-decomposition of A and computing $|A|$ and $A^{-1}B$. Then, for each (α, β, γ), we compute $H = D - C \cdot (A^{-1}B)$ and the 8×8 symbolic determinant $|H + uI|$.

5.3.2 Sequence evaluation

Stürm sequences can be evaluated by constructing the actual sequence of polynomials $\{f(u), f'(u), f_3(u), \ldots\}$ where $f_i = GCD(f_{i-1}, f_{i-2})$, and counting the sign changes in the sequence of constant terms. But the *subresultant polynomial remainder sequence* algorithm [BT71] avoids the exponential coefficient growth in the polynomial GCD, and tends to be more efficient, especially for high-degree polynomials. The subresultant algorithm computes the coefficients of the sequence's polynomials as determinants of matrices. For root-counting, we need only the signs of the constant terms of the polynomials in the sequence. The problem is thus reduced to computing the signs of determinants of matrices with rational entries, or equivalently, integer entries.

The determinant-sign problem has been well-studied in computational geometry ([BEPP97], [BY97], [Cla92]), but most approaches are efficient only for small matrices (order less than 10 and entries of fewer than 53 bits). We find the fastest general-purpose algorithm to be Gaussian elimination over a series of finite fields, followed by reconstruction in accordance with the Chinese remainder theorem. We use the implementation in the software package LiDIA [Gro97]. The finite-field algorithm is protected by a floating-point filter, based on an idea by James Demmel and implemented using the LAPACK library [Dem89]. The filter computes the singular-value decomposition of the matrix in double-precision floating-point, extracts the determinant sign from the decomposition, and examines the singular values to decide whether the sign is dependable. The filter runs in about one-tenth the time of the Chinese remainder algorithm, and for low-degree polynomials, it filters 80–100% of the matrices.

Task	*Time (sec)*
Set up system	1.25
Evaluate at one point	.0360
Count roots in one box	$1.25 + 8 \times .0360 =$ 1.54

Figure 7: *Timing for 3-D Stürm operations*

	Ex. 1	Ex. 2	Ex. 3	Ex. 4
Number of turning points	1	3	2	2
Time to find turning points (sec) (uses 2-D Stürm)	43.0	101	67.7	69.0
Number of edge points	2	6	4	4
Time to find edge points (sec) (uses 1-D Stürm)	0.1	0.1	0.09	0.09
Time to run topology recursion (uses 1-D Stürm)	0.03	0.18	0.13	0.14

Figure 8: *Timing for curve topology*

5.4 Performance of kernel operations

Figure 7 presents a timing measurement from a representative 3-D Stürm computation. The program was run on a 400MHz Pentium computer. The time to set up the system includes the time to determine the Macaulay denominator and the portion of the Macaulay numerator which is not dependent on the evaluation point. This needs to be done only one time per system of equations. To count the number of roots in a box, 8 corner points are tested. Splitting the box in half requires 4 more evaluations; in quarters, 10 more; in eighths, 19 more.

Figure 8 presents timing results from the curve topology algorithm applied to four typical examples. The computer used for these timings is based on a 200MHz Mips R4400 processor. The cases listed are representative curves that arise in the medial axis computation. Each of the curves is has total degree four.

6 Handling Degeneracies

We hold that the degeneracies that arise in the medial axis problem are of only a few types, and may be detected and dealt with explicitly during the course of the algorithm. This is to be contrasted with, for example, the problem of CSG boundary evaluation (see [Kri97] for a treatment) in which one may be faced with a great variety of degenerate situations, some of which have no simple resolution. Only two forms of degeneracy pose a problem to the searching algorithm: the degenerate junction and the degenerate seam. A sheet cannot be equidistant from three boundary elements. If it were, two of the elements must have coincident "flats"—points, lines, or planes—and these two elements' Voronoi regions will be separated by at least one intervening vertex or edge.

In this section we describe the two ways in which degeneracy is manifested in the medial axis problem, and propose means for dealing with these situations.

6.1 The degenerate junction

A junction is said to be degenerate when it is equidistant from five or more boundary elements. Our algorithm detects degenerate junction points by a three-stage process.

1. Discover, during the searching step, that the interval or box containing the starting point overlaps with that of a candidate end point, even after the intervals or boxes have been reduced to a reasonably small size.
2. Check whether the four bisector surfaces involved actually meet at some point, using a Dixon resultant [Dix10].
3. If so, either further refine the root to the precision recommended by Canny's gap theorem [Can88], or check for a root of the system

$$\{f_1^2 + f_2^2 + f_3^2 + f_4^2, f_1, f_2\}$$

   inside the overlap of the boxes.

The first two steps may be regarded as filters for avoiding step 3 where possible.

The algorithm for searching away from a degenerate junction is given in full generality in section 4.3.

6.2 The degenerate seam

A seam is degenerate when it is governed by four or more boundary elements. Our algorithm does not need to do extra work to discover degenerate seams. Such a seam is analyzed using three of its governors, while a fourth governor is necessarily discovered during the search for the ending junction. While attempting to isolate a zero-dimensional solution to this system, the algorithm encounters a Stürm volume function that is identically zero, meaning that the solutions set is infinite. When this occurs, the extra governor is simply added to the list of seam governors.

7 Implementation and Performance

In this section, we describe implementation of our algorithm and highlight its performance on two polyhedra.

7.1 Implementation

The kernel operations have been fully implemented in C++. Routines for root isolation in one, two, and three dimensions take advantage of the efficiency techniques described in sections 5.2 and 5.3. The curve topology algorithm is also fully implemented. Our implementation is based on the data structures for exact arithmetic in Lidia [Gro97] and also takes advantages of routines from LAPACK [Dem89]. The current version of the geometric kernel consists of about 20,000 lines of C++ code.

The high-level, non-time-critical part of the system is in prototype stage, and is partially implemented in about 3,000 lines of Mathematica code. The Mathematica code has not been interfaced with the C++ implementation. Currently our system has been used to compute medial axes of small polyhedra composed of up to about 20 faces.

7.2 Results

Figures 9 and 10 show a simple non-convex polyhedron (two tetrahedra attached at a face) and its medial axis. The medial surfaces are parts of planes and cones, and the seams are segments of lines and ellipses.

A more complex example is given in figures 12 through 15. This polyhedron is an octagonal box with two opposing slots. The two non-convex edges are separated by a very small gap. Visualizing the entire medial axis is difficult. We show a schematic diagram, figure 13, that gives some idea of its geometric complexity. We have substituted straight line segments for the seams. Figure 14 magnifies the schematic in the interesting central region where the non-convex edges approach each other. Finally, the hyperbolic paraboloid that is the bisector of these two edges is shown in figure 15. The seams in the last figure are hyperbolas; the medial axis also contains linear and elliptical seams. In fact, the elliptical seams in both examples are treated as quartics, since they lie in planes with irrational coefficients.

8 Conclusions and Future Work

In this paper, we have presented an accurate algorithm to compute the medial axis of 3-D polyhedron using exact arithmetic. We have also highlighted a number of techniques to improve its efficiency, and demonstrated its performance on a few polyhedra. We are currently working on a C++ implementation of the higher level routines so as to develop a complete and automatic medial axis implementation. In our future work, we plan to investigate use of perturbation techniques to handle degeneracies, and to apply the system to complex polyhedra composed of hundreds of boundary features.

References

[AAS97] P. Agarwal, B. Aronov, and M. Sharir. Computing envelopes in four dimensions with applications. *SIAM J. Comput.*, 26:1714–1732, 1997.

[AB88] S.S. Abhyankar and C. Bajaj. Automatic parametrizations of rational curves and surfaces iii: Algebraic plane curves. *Computer Aided Geometric Design*, 5:309–321, 1988.

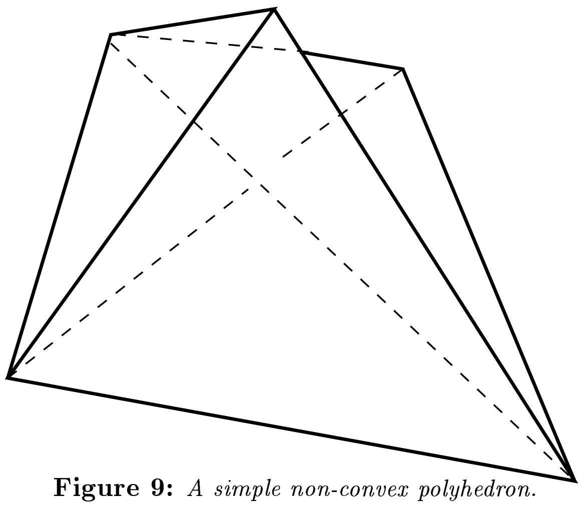

Figure 9: *A simple non-convex polyhedron.*

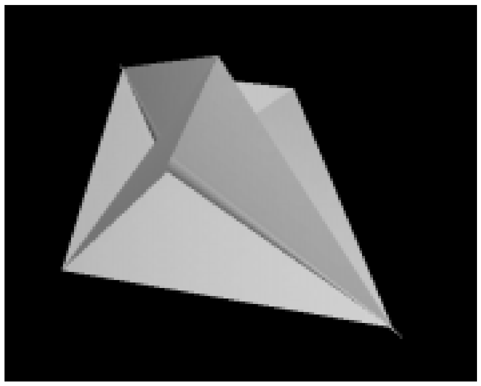

Figure 10: *The medial axis.*

.

Polyhedron	*Sheets*	*Seams*	*Junctions*
Ex. 1 (5 faces)	15	13	4
Ex. 2 (16 faces)	38	68	44

Figure 11: *Two example polyhedra. (Only internal junctions are counted.)*

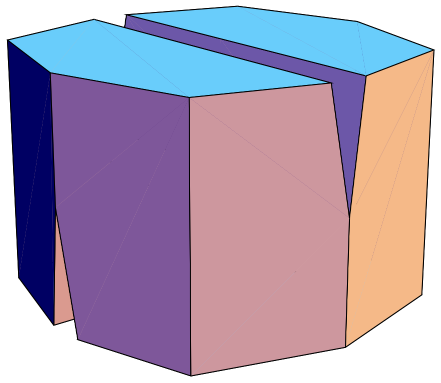

Figure 12: *A polyhedron with two non-convex edges coming very close together.*

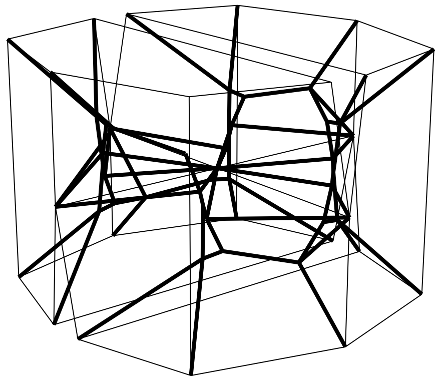

Figure 13: *A schematic of the medial axis, with line segments representing junctions.*

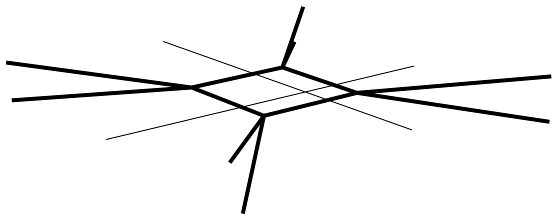

Figure 14: *A* 100× *magnification of the central region of the schematic.*

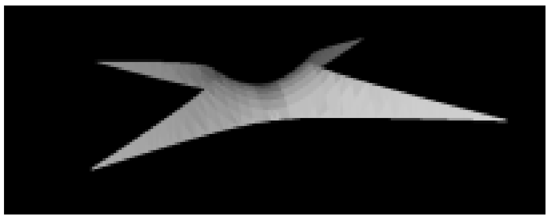

Figure 15: *The bisector of the two non-convex edges, rendered as a trimmed quadric.*

[AF88] S. Arnborg and H. Feng. Algebraic decomposition of regular curves. *J. Symbolic Comput.*, 15(1):131–140, 1988.

[BEPP97] H. Bronnimann, I. Emiris, V. Pan, and S. Pion. Computing exact geometric predicates using modular arithmetic with single precision. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 174–182, 1997.

[Blu67] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, 1967.

[BMP94] M. Benouamer, D. Michelucci, and B. Peroche. Error-free boundary evaluation based on a lazy rational arithmetic: a detailed implementation. *Computer-Aided Design*, 26(6), 1994.

[Bra92] J. W. Brandt. Describing a solid with the three-dimensional skeleton. In J. D. Warren, editor, *Proceedings of the International Society for Optical Engineering Volume 1830, Curves and Surfaces in Computer Vision and Graphics III*, pages 258–269. SPIE, Boston, 1992.

[BT71] J.S. Brown and J.F. Traub. On euclid's algorithm and the theory of subresultant. *Journal of ACM*, 18(4):505–514, 1971.

[BY97] H. Bronnimann and M. Yvinec. Efficient exact evaluation of signs of determinants. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 166–173, 1997.

[Can88] J.F. Canny. *The Complexity of Robot Motion Planning*. ACM Doctoral Dissertation Award. MIT Press, 1988.

[Chi92] C.-S. Chiang. *The Euclidean distance transform*. Ph.D. thesis, Dept. Comput. Sci., Purdue Univ., West Lafayette, IN, August 1992. Report CSD-TR 92-050.

[Cla92] K. L. Clarkson. Safe and effective determinant evaluation. In *Proc. 33rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 387–395, 1992.

[Dem89] J. Demmel. LAPACK: A portable linear algebra library for supercomputers. In *Proceedings of the 1989 IEEE Control Systems Society Workshop on Computer-Aided Control System Design*, Tampa, FL, Dec 1989. IEEE.

[DH90] D. Dutta and C. M. Hoffmann. A geometric investigation of the skeleton of CSG objects. In *Proc. ASME Conf. Design Automation*, 1990.

[DH93] D. Dutta and C. M. Hoffmann. On the skeleton of simple CSG objects. *Journal of Mechanical Design, ASME Transactions*, 115(1):87–94, 1993.

[Dix10] A. L. Dixon. The eliminant of the equations of four quadric surfaces. *Proc. London Math. Soc.*, pages 340–352, 1910.

[EK96] G. Elber and M-S. Kim. The bisector surface of freeform rational space curves. Technical Report CIS Report #9619, Technion-Israel Institute of Technology, September 1996.

[For87] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[For95] S. Fortune. Polyhedral modeling with exact arithmetic. *Proceedings of ACM Solid Modeling*, pages 225–234, 1995.

[Gro97] LiDIA Group. A library for computational number theory. Technical report, TH Darmstadt, Fachbereich Informatik, Institut fur Theoretische Informatik, 1997.

[Hel97] Martin Held. Voronoi diagrams and offset curves of curvilinear polygons. *Computer–Aided Design*, 1997. To appear.

[Hof89] C.M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, California, 1989.

[Hof94] C. M. Hoffmann. How to construct the skeleton of csg objects. In A. Bowyer and J. Davenport, editors, *Proceedings of the Fourth IMA Conference, The Mathematics of Surfaces, University of Bath, UK, September 1990.* Oxford University Press, New York, 1994.

[Ima96] T. Imai. A topology oriented algorithm for the Voronoi diagram of polygons. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 107–112. Carleton University Press, Ottawa, Canada, 1996.

[Joh87] J.K. Johnstone. *The Sorting of points along an algebraic curve.* PhD thesis, Cornell University, Department of Computer Science, 1987.

[KKM97] J. Keyser, S. Krishnan, and D. Manocha. Efficient and accurate b-rep generation of low degree sculptured solids using exact arithmetic. In *ACM/SIGGRAPH Symposium on Solid Modeling*, pages 42–55, 1997.

[Kri97] Shankar Krishnan. *Efficient and Accurate Boundary Evaluation Algorithms for Boolean Combinations of Sculptured Solids.* PhD thesis, University of North Carolina-Chapel Hill, 1997.

[Lee82] D. T. Lee. Medial axis transformation of a planar shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-4:363–369, 1982.

[Mac02] F.S. Macaulay. On some formula in elimination. *Proceedings of London Mathematical Society*, 1(33):3–27, May 1902.

[Mil92] P. S. Milne. On the solutions of a set of polynomial equations. In *Symbolic and Numerical Computation for Artificial Intelligence*, pages 89–102, 1992.

[Mil93] V. Milenkovic. Robust construction of the Voronoi diagram of a polyhedron. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 473–478, 1993.

[Ped91] P. Pedersen. Multivariate sturm theory. In *Proceedings of AAECC*, pages 318–332. Springer-Verlag, 1991.

[RT95] Jayachandra Reddy and George Turkiyyah. Computation of 3D skeletons using a generalized Delaunay triangulation technique. *Computer–Aided Design*, 27(9):677–694, 1995.

[SAR95] D. J. Sheehy, C. G. Armstrong, and D. J. Robinson. Computing the medial surface of a solid from a domain Delaunay triangulation. In *Proc. ACM/IEEE Symp. on Solid Modeling and Applications*, may 1995.

[Sei90] R. Seidel. Linear programming and convex hulls made easy. In *Proc. 6th Ann. ACM Conf. on Computational Geometry*, pages 211–215, Berkeley, California, 1990.

[SI89] K. Sugihara and M. Iri. A solid modeling system free from topological inconsistencies. *J. Inf. Proc., Inf. Proc. Soc. of Japan*, 12(4):380–393, 1989.

[SNTM92] V. Srinivasan, L. R. Nackman, J.-M. Tang, and S. N. Meshkat. Automatic mesh generation using the symmetric axis transform of polygonal domains. *Proc. IEEE*, 80(9):1485–1501, September 1992.

[SPB95] Evan C. Sherbrooke, Nicholas M. Patrikalakis, and Erik Brisson. Computation of the medial axis transform of 3-D polyhedra. In *Solid Modeling*, pages 187–199. ACM, 1995.

[Ver94] P. J. Vermeer. *Medial Axis Transform to Boundary Representation Conversion.* PhD thesis, CS Dept., Purdue University, West Lafayette, Indiana 47907-1398, USA, 1994.

[VO95] Jules Vleugels and Mark Overmars. Approximating generalized Voronoi diagrams in any dimension. Technical Report UU-CS-1995-14, Department of Computer Science, Utrecht University, 1995.

[Wol92] F. E. Wolter. Cut locus and medial axis in global shape interrogation and representation. *Computer Aided Geometric Design*, 1992.

[Yu92] J. Yu. *Exact arithmetic solid modeling.* PhD thesis, Purdue University, 1992.