Computational geometry for curved objects: Voronoi diagrams in the plane

George M. Tzoumas[*]

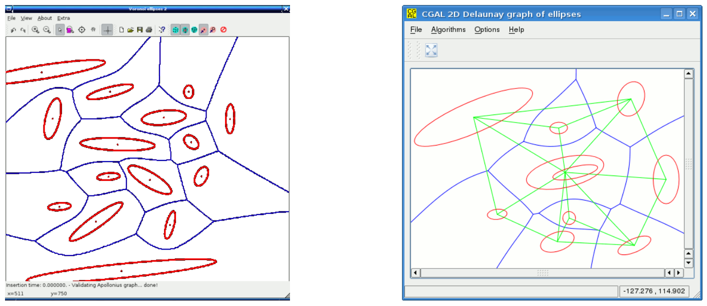
National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
geotz@di.uoa.gr

Abstract. We examine the problem of computing exactly the Delaunay graph (and the dual Voronoi diagram) of a set of, possibly intersecting, smooth convex pseudo-circles in the Euclidean plane, given in parametric form. Pseudo-circles are (convex) closed curves, every pair of which has at most two intersection points. We propose robust end efficient algorithms for all required predicates under the exact computation paradigm, analyzing their algebraic complexity. To speed up the algebraic computations, we exploit geometric properties of the problem and provide a subdivision-based algorithm that exhibits quadratic convergence, allowing for real-time evaluations. Finally, we present a CGAL-based C++ implementation for the case of ellipses, which is, to the best of our knowledge, the first exact implementation in non-linear computational geometry. Our code spends about 98 sec to construct the Delaunay graph of 128 non-intersecting ellipses, when no degeneracies occur. It is faster than the CGAL segment Delaunay graph, when ellipses are approximated by k-gons for $k > 15$.

1 Introduction

Computing the Delaunay graph, and its dual Voronoi diagram, of a set of sites in the plane has been studied extensively due to its numerous applications, including motion planning, assembly, surface reconstruction, and crystallography. In contrast to most existing approaches, our work guarantees the *exactness* of the Delaunay graph. This means that all combinatorial information is correct, namely the definition of the graph's edges, which, of course, have no geometric realization. The dual Voronoi diagram involves algebraic numbers for representing the vertices and bisector edges: our algorithms and software handle these numbers exactly. For instance, they can answer all point-location queries correctly. Hence, we say that our representation of the dual Voronoi diagram adheres to the principles of the *exact computation paradigm*, a distinctive feature in the realm of non-linear computational geometry. For drawing the diagram, our methods allow for an approximation of these numbers with arbitrary precision, and this precision need not be fixed in advance.

[*] Dissertation Advisor: Ioannis Z. Emiris, Professor


Voronoi ellipses 2
File View About Extra
CGAL 2D Delaunay graph of ellipses
File Algorithms Options Help

Insertion time: 0.000000. - Validating Apollonius graph... done!
x=511 y=750

-127.276 , 114.902

Fig. 1. Left: Voronoi diagram of 15 ellipses; Right: Voronoi diagram and Delaunay graph of 10 ellipses

Input sites have usually been linear objects, the hardest cases being line segments and polygons [9,10]; moreover, only the latter yields an exact output. The approximation of smooth curved objects by (non-smooth) linear or circular segments may introduce artifacts and new branches in the Voronoi diagram, thus necessitating post-processing. It may even yield topologically incorrect results, as explained in [14].

The Voronoi diagram has been studied in the case of planar sites with curved boundaries [14], where topological properties are demonstrated, including the type of bisector curves, though the predicates and their implementation are not considered. There are works that compute the planar Voronoi diagram *approximately*: In [8], curve bisectors are traced within machine precision to compute a single Voronoi cell of a set of rational C^1-continuous parametric closed curves. The runtime of their implementation varies between a few seconds and a few minutes. It is briefly argued that the method extends to exact arithmetic, but without elaborating on the underlying algebraic computations or the handling of degeneracies. In another work [15], the boundary of the sites is traced with a prescribed precision, while [13] suggests working with lower-degree approximations of bisectors of curved sites.

Few works have studied *exact* Delaunay graphs for curved objects. In the case of circles, the exact and efficient implementation of [5] is now part of CGAL.[1] Conics were considered in [1], but only in a purely theoretical framework. Moreover, the algebraic conditions derived were not optimal, leading to a prohibitively high algebraic complexity.

In [11], the authors study the properties of smooth convex, possibly intersecting, pseudo-circles. They show that the Voronoi diagram of these sites belongs to the class of *abstract Voronoi diagrams* [12] and propose an incremental algorithm that relies on certain geometric predicates. The evaluation of the predicates themselves is not considered, this is the problem addressed by this thesis.

[1] http://www.cgal.org/

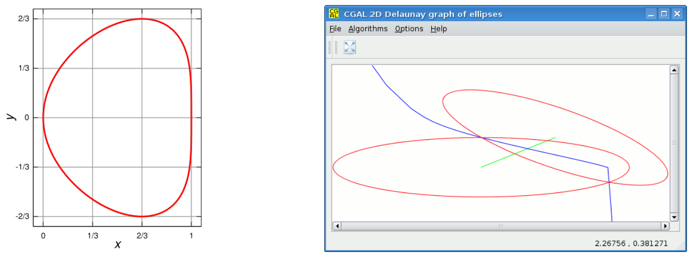


Fig. 2. Left: The Bean curve $t \mapsto (\frac{1+t^2}{t^4+t^2+1}, \frac{t(1+t^2)}{t^4+t^2+1})$; Right: Voronoi diagram of two intersecting ellipses

2 Preliminaries

Our input is smooth convex closed curves given in parametric form. For an example of such a curve we refer the reader to fig. 2 left. Smoothness allows the tangent (and normal) line at any point of the curve to be well-defined. We denote by C_t a smooth closed convex curve parametrized by t. We refer to a point p on C_t with parameter value $\hat{t}$ by $p_{\hat{t}}$, or simply by $\hat{t}$, when it is clear from context. By $C_t{}^\circ$ we denote the interior of the region bounded by the curve C_t. $\mathbf{C_t}$ is a smooth convex object (site), so that if p denotes a point in the plane, $p \in \mathbf{C_t} \iff p \in C_t \cup C_t{}^\circ$. When two sites intersect, we assume that their boundaries have at most two intersections, i.e. they form *pseudo-circles.* A curve C_t is given by the map

$$C_t : \mathbb{R} \ni t \mapsto (X_t(t), Y_t(t)) = \left(\frac{F_t(t)}{H_t(t)}, \frac{G_t(t)}{H_t(t)}\right), \tag{1}$$

but actual denominators *can differ;* we use (1) for simplicity in our proofs.

Here F_t, G_t and H_t are polynomials in $\mathbb{Z}[t]$, with degrees bounded by d. All algorithms, predicates and the corresponding analysis are valid for any parametric curve, even when the polynomials have different degrees, though we use (1) for simplicity. We assume that $H_t(t) \neq 0 \forall t \in \mathbb{R}$. To simplify notation we write F_t instead of $F_t(t)$, and denote its derivative with respect to t as F_t'. When $d = 2$ the curves defined are conics: ellipses and circles are the only closed convex curves represented.

Let $p_t(X_t, Y_t)$ be a point on the curve C_t. The equation of the line that supports the normal at p_t is $(N_t) : (x - X_t)X_t' + (y - Y_t)Y_t' = 0$, which is a polynomial $N_t(x, y, t) \in \mathbb{Z}[x, y, t]$, linear in x and y and of degree $\leq 3d - 2$ in t .

Given a curve C_t, if we consider rationals $t_1 \neq t_2$ in $\mathbb{R}$, then the point $((X_t(t_1) + X_t(t_2))/2, (Y_t(t_1) + Y_t(t_2))/2)$ belongs to $C_t{}^\circ$; such a point is p in fig. 3. To characterize the relative position of sites $\mathbf{C_t}$, $\mathbf{C_r}$, we compute and characterize (as external or internal) all their bitangent lines. For example, if two sites do not intersect, then they have 2 internal and 2 external bitangents. The same technique decides the relative position of a point and a site since, i.e.,

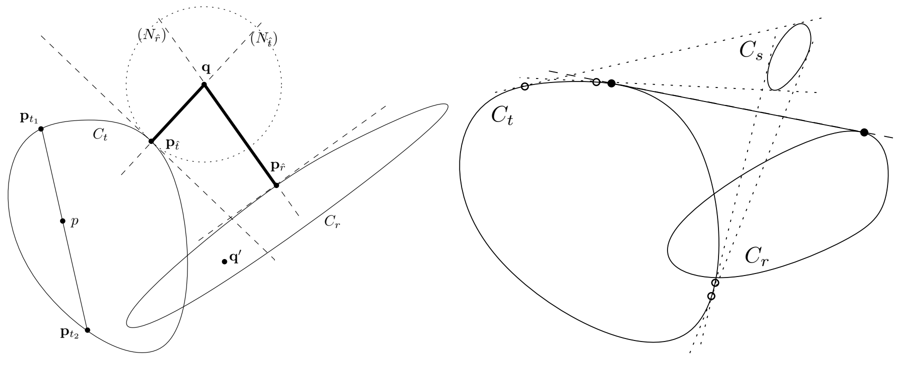


Fig. 3. Deciding SideOfBisector (left) and DistanceFromBitangent (right)

when a point is interior to a site there are no supporting lines tangent to the site.

3 Basic predicates

In this section we examine the predicates for the incremental algorithm of [11]. The expected complexity of an insertion (in a diagram with n sites) is $\mathcal{O}(\log^2 n)$ for disjoint sites. Computing an exact Delaunay graph implies that we identify correctly all degenerate cases, including Voronoi circles tangent to more than 3 sites. The insertion of a new site consists roughly of the following: (i) Locate the nearest neighbor of the new site, (ii) Find a conflict between an edge of the current diagram and the new site, or detect that the latter is internal (hidden) in another site, in which case it does not affect the Delaunay graph nor the Voronoi diagram. (iii) Find the entire *conflict region,* defined as that part of the Voronoi diagram which changes due to the insertion of the new site, and update the dual Delaunay graph. In the sequel, we analyze the predicates needed for the above three steps. However, predicate InCircle is presented separately due to its higher complexity.

We shall now present the SideOfBisector predicate. First, we recall from [11] the definition of distance. Given a site $\mathbf{C_t}$ and a point q in the plane, the (signed) distance $\delta(q, \mathbf{C_t})$ from q to $\mathbf{C_t}$ equals $min_{x \in C_t} ||q - x||$ when $q \notin \mathbf{C_t}$ and to $-min_{x \in C_t} ||q - x||$ when $q \in \mathbf{C_t}$, where $|| \cdot ||$ denotes the Euclidean norm. The absolute value of the distance equals the radius of the smallest circle centered at q tangent to $\mathbf{C_t}$. Given two sites $\mathbf{C_t}$ and $\mathbf{C_r}$ and a point $q = (q_1, q_2) \in \mathbb{Q}^2$, this predicate decides which site is closest to the point. If $q \notin \mathbf{C_t}$ and $q \in \mathbf{C_r}$, then q is closer to $\mathbf{C_r}$. For example, in fig. 3, q' is closer to $\mathbf{C_r}$. Otherwise, if q lies outside or inside both sites, we have to compare the distances from q to the two curves. To find the site closest to q it suffices to compare the (squared) lengths of segments $qp_{\hat t}$ and $qp_{\hat r}$. We can express this length as an algebraic number of degree $\leq 3d-2$. Therefore this predicate is decided by comparing two such algebraic numbers [6]. An alternative approach for ellipses is to consider the *pencil* of two conics [4].

Now we consider the DistanceFromBitangent predicate. Consider two sites, $\mathbf{C_t}$ and $\mathbf{C_r}$, and their CCW bitangent line, which leaves both sites on the right hand-side, as we move from the tangency point of C_t to the tangency point of C_r; such a bitangent appears in fig. 3. This line divides the plane into two halfplanes and DistanceFromBitangent (abbreviated by DFB from now on) decides whether a third site, $\mathbf{C_s}$, lies in the same halfplane as the other two.

We split the problem into two sub-problems. The first consists in computing the external bitangent of interest, while the second consists in deciding the relative position of the third site with respect to this bitangent. The bitangents of C_t and C_r are modeled through the vanishing of a discriminant of a polynomial. The degree of the discriminant is $\leq 4(d-1)^2$ and its solution yields the tangency points of the bitangent lines. To decide the position of $\mathbf{C_s}$ w.r.t. the CCW bitangent line, we consider the tangency points of all the bitangents of C_t and C_s, shown with circular marks in fig. 3. We can then decide the position of C_s by the ordering of the aforementioned points *and* the tangency point of the bitangent and C_t, shown with solid circular mark in the same figure [4,6].

4 InCircle

This section introduces a polynomial system for expressing the Voronoi circle, leading to a robust and fast implementation of our main predicate. Recall that the Voronoi circle is centered at a Voronoi vertex whose radius equals the distance between the vertex and the sites closest to it (i.e., the circle is tangent to the sites). A Voronoi disk is a disk defined by a Voronoi circle.

Given sites $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$ in this order, we denote their associated Voronoi disk by V_{trs} iff their tangency points on the disk are in CCW direction (cf. fig. 6 middle). In this case, V_{trs} is a CCW Voronoi disk, and V_{tsr} is a CW Voronoi disk. Since the Voronoi diagram of smooth convex pseudo-circles is an abstract Voronoi diagram, given 3 sites, there may exist at most one CCW Voronoi disk and at most one CW Voronoi disk. Moreover, these disks may be either external (externally tangent to the sites) or internal (internally tangent).

Let us now adapt the definition of conflict from [11], (see also fig. 4):

Definition 1. *Given sites* $\mathbf{C_t}$*,* $\mathbf{C_r}$*,* $\mathbf{C_s}$*, let* V_{trs} *be their Voronoi disk and* $\mathbf{C_h}$ *be a query site. If* V_{trs} *is an external Voronoi disk, then* $\mathbf{C_h}$ *is in conflict with* V_{trs} *iff* V_{trs} *is intersecting* ${C_h}^{\circ}$*. If* V_{trs} *is an internal Voronoi disk, then* $\mathbf{C_h}$ *is in conflict with* V_{trs} *iff* V_{trs} *is included in* ${C_h}^{\circ}$*.*

Given $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$, InCircle decides if a newly inserted site $\mathbf{C_h}$ is in *conflict* with V_{trs}. A degeneracy arises when $\mathbf{C_h}$ is also tangent to V_{trs}. Given that V_{trs} exists, the predicate is computed as follows: (i) Solve the algebraic system that expresses the Voronoi circle. Among the solutions (which correspond to various tritangent circles, cf. fig. 6 middle), find V_{trs}. (ii) Determine the relative position of $\mathbf{C_h}$ w.r.t. V_{trs}.

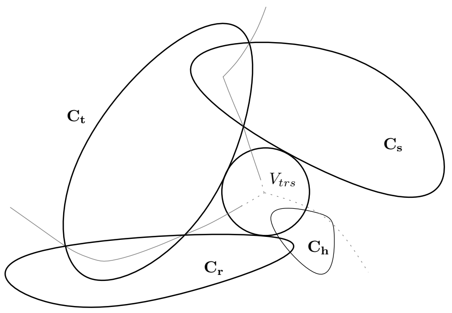


Fig. 4. Conflict query site $\mathbf{C_h}$ with the external Voronoi disk V_{trs}

Bitangent circles. Given two sites $\mathbf{C_t}$ and $\mathbf{C_r}$, their *bisector* is the locus of points that are equidistant from the two sites. Let $q(x, y)$ be a point on the bisector. Then q is the intersection of three lines, namely the normal lines from q to each site, and the bisector line of the two footpoints (a subset of fig. 6 left) which is expressed by the following system.

$$N_t(x, y, t) = N_r(x, y, r) = M_{tr}(x, y, t, r) = 0. \tag{2}$$

Eliminating x, y we obtain a bivariate polynomial $B(t, r)$ of degree $\leq 4d - 2$ in each parameter [3]. Note that for the case of ellipses, that concerns our implementation, the implicit form of the bisector curve is of total degree 28, while the parametric one is of total degree 12. Each point q on the bisector is the center of a bitangent circle to the two sites. The algebraic curve $B(t, r)$ contains some branches that do not correspond with the definition of our distance (since the *min* function is involved). We are interested only in the externally bitangent circles, or the internally bitangent ones (we call such circles *Apollonius* circles of two sites).

We pick the proper branch of the bisector exploiting the following geometric properties we have discovered. Consider a point $\hat{t} \in C_t$. $B(\hat{t}, r)$ is a univariate polynomial and its solutions are the tangency points of bitangent circles on C_r (with fixed tangency point $\hat{t}$ on C_t). Let $\hat{r}$ be the tangency point on C_r. $B(\hat{t}, \hat{r})$, abbreviated as $\hat{B}$, corresponds to a *specific* bitangent circle. If $\hat{t} \in {C_r}^{\circ}$, then $\hat{B}$ should correspond to an *internal* bitangent disk. Therefore, we have to make sure that its radius is bounded by that of the maximal disk tangent at $\hat{t}$ and included in $\mathbf{C_t}$. Note that the center of such a maximal disk lies on the *medial axis* of $\mathbf{C_t}$. If $t \notin {C_r}^{\circ}$ (fig. 6 right), then $\hat{B}$ should correspond to an *external* bitangent disk. Therefore $\hat{t}$ has to lie in the interior of the convex hull (CH) of $\mathbf{C_t}$ and $\mathbf{C_r}$. Moreover, $\hat{r}$ lies in an arc, whose endpoints are the tangency points of the tangent lines from $\hat{t}$ to C_r. If the tangent line ϵ at $\hat{t}$ intersects C_r, then one endpoint of the arc is replaced by an intersection point of ϵ and C_r.

Computing Voronoi circle by subdivision. In the parametric space, the intersection of two bisectors involves three variables. In order to express the Voronoi

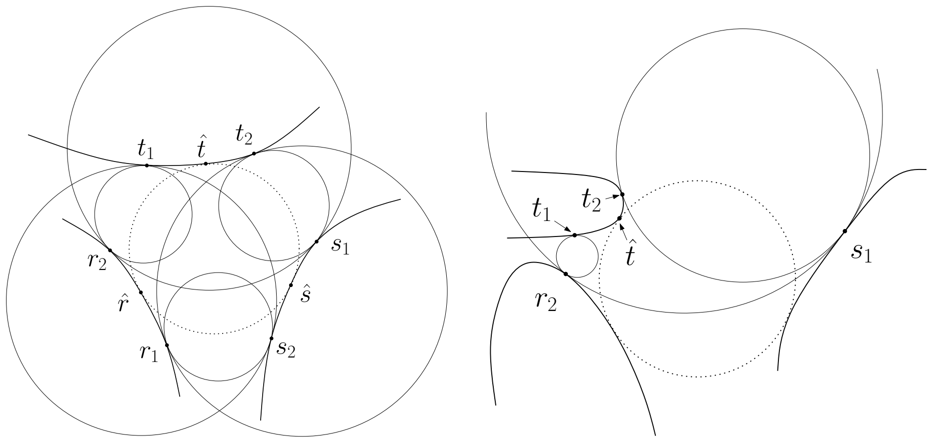


Fig. 5. Left: All-pair bitangent circles. Right: t_2, when computed from t_1, encloses $\hat{t}$

circle, we consider the intersection of three bisectors by solving the system:

$$B_1(t, r) = B_2(s, t) = B_3(r, s) = 0. \tag{3}$$

The basic idea of our algorithm [7] is the following: Let $(\hat{t}, \hat{r}, \hat{s})$ be the solution of (3) we are looking for. Now consider the following system:

$$B_1(t_1, r_2) = B_3(r_2, s_1) = B_2(s_1, t_2) = 0 \tag{4}$$

$$B_1(t_2, r_1) = B_3(r_1, s_2) = B_2(s_2, t_1) = 0 \tag{5}$$

These two systems look like (3). The difference is that we have considered $t_1 \neq t_2$ in the general case and thus we can start solving the above equations in the given order. Doing so and keeping solutions that correspond to Apollonius circles (using the geometric arguments described previously), leads to a construction as in fig. 5 left. All bitangent circles coincide with the Voronoi circle when $t_1 = \hat{t} = t_2$. Otherwise, we have found an interval $[t_1, t_2]$ that contains $\hat{t}$. We can refine it by choosing a new point t'_1 inside this interval and computing $[t'_1, t'_2]$ (suppose without any harm that $t'_1 < t'_2$). It holds that t_1 approaches $\hat{t}$ from the left $\Rightarrow r_2$ approaches $\hat{r}$ from the right $\Rightarrow s_1$ approaches $\hat{s}$ from the left $\Rightarrow t_2$ approaches $\hat{t}$ from the right (see fig. 5 right). Therefore $t'_1 \in [t_1, t_2] \Rightarrow \hat{t} \in [t'_1, t'_2] \subset [t_1, t_2]$. Note that computing a smaller interval on dimension t allows us to compute smaller intervals on dimensions r and s as well. Therefore we maintain exactly one box that contains our solution, contrary to generic interval-arithmetic techniques that may need to maintain a large number of boxes.

Theorem 1. *The above subdivision-based algorithm converges quadratically.*

Proof. By using the *implicit function theorem* and Taylor expansions [7].

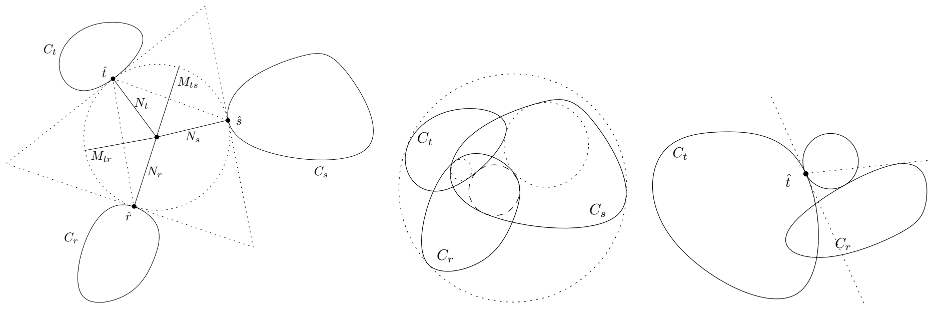


Fig. 6. Left: An external tritangent circle; Middle: Various tritangent circles. Dotted line: CCW(t,r,s), Dashed line: CW(t,r,s); Right: Finding an external bitangent circle

Computing Voronoi circle algebraically. The polynomial system expressing all circles tangent to $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$ is:

$$N_t(x,y,t) = N_r(x,y,r) = N_s(x,y,s) = M_{tr}(x,y,t,r) = M_{ts}(x,y,t,s) = 0. \quad (6)$$

The first 3 equations correspond to normals at points t, r, s on the 3 given sites. All normals go through the Voronoi vertex (x, y). The last two equations force (x, y) to be equidistant from the sites: each one corresponds to the bisector of the segment between two footpoints (cf. fig. 6 left). This system was also used in [13], but solved with iterative methods. Elimination of x, y from M_{tr}, N_t, N_r yields the *bisector* of two sites w.r.t. t, r. It turns out that the resultant of (6) can be computed faster than that of (2), as the former system has simpler redundant solutions. However, (2) is more appropriate for the subdivision scheme, as it contains only two variables per equation.

Solving the system. The *resultant* of $n + 1$ polynomials in n variables is an irreducible[2] polynomial in the coefficients of the polynomials which vanishes iff the system has a complex solution. In particular, sparse (or toric) resultants express the existence of solutions in $(\mathbb{C}^*)^n$ [2]. It is impossible to compute the resultant of 5 arbitrary polynomials as a determinant, so we apply successive Sylvester determinants, i.e., optimal resultant formulae for $n = 1$. This typically produces extraneous factors but, by exploiting the fact that some polynomials are linear, and that none contains all variables, we shall provide the *complete* factorization of the computed polynomial; we focus on conics for simplicity, but our approach holds for any parametric curve. We denote by $\Pi(t)$ the resultant of (6) when eliminating all variables except t: it is, generally, an irreducible univariate polynomial and vanishes at the values of t that correspond to the complex tritangent circles. Recall that the curves are defined by (1).

[2] Irreducibility occurs for generic coefficients; otherwise, resultants can be factorized.

Theorem 2. *If $\Pi(t)$ is the resultant of (6) as above, then* $\mathsf{Res}_{xy}(R_1, R_2, N_t) = \Pi(t)H_t^{40}(G_tH_t' - G_t'H_t)^{36}$*, where,* $R_1 = \mathsf{Res}_r(M_{tr}, N_r)$*,* $R_2 = \mathsf{Res}_s(M_{ts}, N_s)$*, and the degree of Π is 184.*

The above theorem provides an upper bound of 184 complex tritangent circles to 3 conics. Numeric examples show that the bound is tight.

Corollary 1. *The degree of the resultant of (6) for general parametric curves, as in (1), is bounded by* $(3d-2)(5d-2)(9d-2)$*, after dividing out the factor of* $(H_t(G_tH_t' - G_t'H_t))^{(5d-2)^2}$*.*

A more careful analysis may exploit term cancellations to yield a tighter bound.

If we solve the resultant of system (6), we obtain one coordinate of the solution vectors (in isolating interval representation). There are methods to obtain the other variables, too. For instance, plugging a value of t in the bisector equation of sites C_t and C_r allows us to find the corresponding value for r. In practice, we are using our subdivision scheme to identify the box that contains the right solutions.

Moreover, having obtained the resultant allows us to detect *degenerate* configurations, i.e., all 4 sites being tangent to the same Voronoi disk. Consider the triplets $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_s}$ and $\mathbf{C_t}$, $\mathbf{C_r}$, $\mathbf{C_h}$. Let $\Pi_1(t), \Pi_2(t)$ be the resultants of (6) for these triplets respectively. The triplets admit an identical tritangent circle iff $gcd(\Pi_1, \Pi_2) \neq 1$. Checking that the common tritangent circle corresponds to the Voronoi circle (i.e., internal or external tritangent one) can be verified by looking at the other coordinates, through our subdivision scheme. We can state the following theorem:

Theorem 3. *Comparing the real roots of $\Pi_1(t)$ with those of $\Pi_2(t)$ allows us to decide* InCircle*. In the case of ellipses, this translates to comparisons of algebraic numbers of degree 184.*

5 Implementation & experiments

We shall briefly describe our efficient and exact implementation for non-intersecting ellipses in the plane (fig. 1 left), which is now being extended to handle pseudo-circles (fig. 2 right), or sites fully contained in other ones (fig. 1 right). Our code is based on the existing CGAL Apollonius package for the combinatorial part of the algorithm. Since CGAL follows the generic programming paradigm, the main issue was to implement the predicates for ellipses, generalizing the circular sites developed for the Apollonius diagram.

For the required algebraic operations, we relied on SYNAPS, an algebraic library which features state-of-the-art implementations for real solving, used to solve the degree-184 univariate polynomials.

We have implemented polynomial interpolation to compute the resultant of system (6), applying thm. 2 at the same time. To speed up the implementation, we use NTL,which is an open source C++ library providing asymptotically fast algorithms for polynomial GCD and Sylvester resultants.

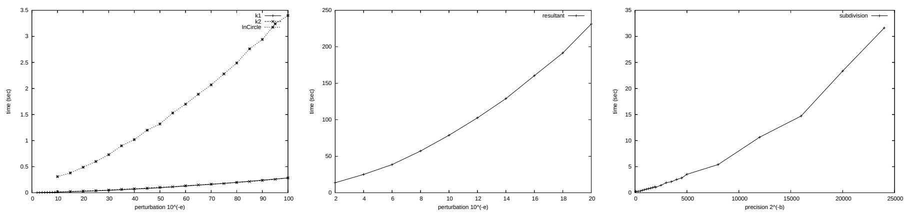


Fig. 7. Left: Predicates; Middle: Resultant computation; Right: Subdivision

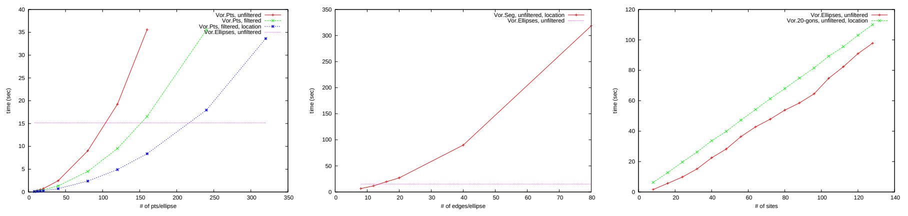


Fig. 8. Delaunay graph of: 32 ellipses vs point approximations with increasing number of points per ellipse (left), 32 polygons with increasing number of edges (middle), polygons and ellipses (right)

For InCircle, we have implemented our specialized subdivision method using interval arithmetic provided by synaps and multi-precision floating point numbers by mpfr. This is a filter, which answers InCircle before full precision is achieved at non-degenerate cases. When the mpfr precision is not enough, we fall back to the exact algebraic method.

Overall, our software design, is generic so as to distinguish the geometry from the algebra part, allowing us to try different algebraic libraries. Currently, we are also using cgal's Algebraic Kernel which provides univariate real solving, multivariate polynomial handling and resultant computation via interpolation.

Now, we present various experimental results. All runtimes are obtained on a Pentium-4 2.6 GHz machine with 1.5GB of RAM, unless otherwise specified.

We have measured the performance of SideOfBisector, DFB and InCircle with varying bitsize. Left fig. 7 corresponds to ellipses with randomly perturbed parameters (axes, rotation and center of ellipses) by adding / subtracting 10^{-e}, with varying e, to small (10-bit) random input parameters; this forces the polynomials computed during each predicate evaluation to have coefficients of large bitsize. All runtimes appear to grow sub-quadratically in e, which is expected since SideOfBisector, DFB have constant arithmetic complexity and InCircle is handled by the subdivision algorithm with quadratic convergence, hence computes τ bits in $\mathcal{O}(\log(\tau))$ steps. In case of degeneracies, the runtime of InCircle is dominated by the resultant computation, shown in middle fig. 7.

Finally, we measured the time needed for the subdivision algorithm to reach a precision of 2^{-b}, using mpfr floats, in right fig. 7. This precision is achieved in about 0.2 sec, and 1 sec suffices for almost 2000 bits of precision, whereas the 24k-bit approximation needs about 0.5 min. This shows that the theoretical separation bound of several million bits [4] cannot be achieved efficiently, hence the usefulness of resultant-based methods. On the other hand, resultants, even with 10-bit input coefficients can be about 70 times slower than the subdivision algorithm using the standard floating point precision of 2^{-53}. In short, both methods have to be combined for a robust and fast solution.

The overall time for the construction of the Delaunay graph (and the structure representing its dual diagram) is shown in right fig. 8 (solid line). It takes, for instance, 98 sec to compute the exact Delaunay graph of 128 non-intersecting ellipses. More importantly, it is about linear in the number of sites for up to this number of non-intersecting ellipses.

Comparing with point approximations. Each ellipse is approximated by a constant number of k points taken uniformly on its boundary (just like the vertices of the polygons in fig. 9, right). These points have rational coordinates, as they are obtained using (1). We compare against 3 variations of the incremental algorithm of CGAL for the Delaunay triangulation: (i) without filtering,(ii) with filtering, (iii) with filtering and improved nearest neighbor location. Left fig. 8 presents results concerning 32 ellipses, with k varying from 8 to 320. We see that the Delaunay graph computation of 32 ellipses is faster for variations (i),(ii),(iii) of the Delaunay triangulation of points for $k \geq 120$, $k \geq 160$ and $k \geq 240$ respectively.

Comparing with polygonal approximations. We compare against the CGAL package for the segment Delaunay graph (and the dual Voronoi diagram) [10]. We replaced each ellipse (fig.9 left) by a 20-gon (fig.9 right). Right fig. 8 shows the time for the incremental construction of the Voronoi diagram of polygons by CGAL (dashed line) compared to that of ellipses (solid line) when the number of sites varies from 4 to 128. Again, nearest neighbor queries are performed smartly. Middle fig. 8 shows the required time to construct the Delaunay graph of 32 ellipses (solid line) and that of 32 polygons approximating each ellipse with a varying number of edges (dotted line). As the number of edges per ellipse increases, the squared-logarithmic cost per insertion becomes non-negligible. Interestingly, the Delaunay graph of polygons is slower with > 15 segments per ellipse.

References

1. F. Anton. *Voronoi diagrams of semi-algebraic sets.* PhD thesis, The University of British Columbia, January 2004.
2. D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry.* Number 185 in Graduate Texts in Math. Springer, New York, 2nd edition, 2005.
3. G. Elber and Myung-Soo Kim. Bisector curves of planar rational curves. *Comp.-Aid. Des.*, 30:1089–1096, 1998.

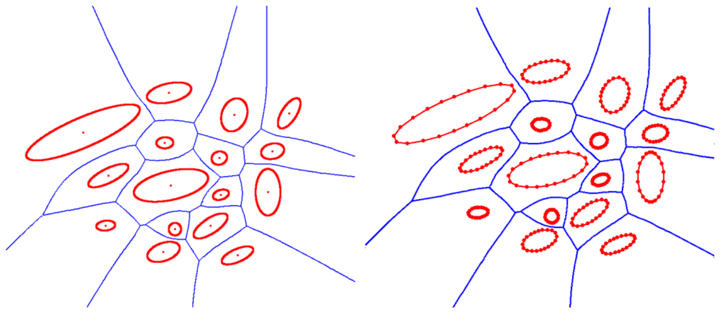


Fig. 9. Left: Voronoi diagram of 16 ellipses. Right: Voronoi diagram of 16 20-gons approximating each ellipse (320 segments in total)

4. I. Z. Emiris, E. P. Tsigaridas, and G. M. Tzoumas. Predicates for the exact Voronoi diagram of ellipses under the euclidean metric. *Intern. J. Comp. Geom. & Appl.*, 18(6):567–597, 2008. Special Issue.
5. I.Z. Emiris and M.I. Karavelas. The predicates of the Apollonius diagram: algorithmic analysis and implementation. *Comp. Geom.: Theory & Appl., Spec. Issue on Robust Geometric Algorithms and their Implementations*, 33(1-2):18–57, 2006.
6. I.Z. Emiris, E.P Tsigaridas, and G.M. Tzoumas. Exact Delaunay graph of smooth convex pseudo-circles: General predicates, and implementation for ellipses. In *Proc. SIAM/ACM Joint Conf. Geometric & Solid Modeling*, San Francisco, October 2009. To appear.
7. I.Z. Emiris and G.M. Tzoumas. Exact and efficient evaluation of the InCircle predicate for parametric ellipses and smooth convex objects. *Comp.-Aid. Des.*, 40(6):691–700, 2008.
8. I. Hanniel, R. Muthuganapathy, G. Elber, and M.-S. Kim. Precise Voronoi cell extraction of free-form rational planar closed curves. In *Proc. 2005 ACM Symp. Solid and Phys. Modeling*, pages 51–59, Cambridge, Massachusetts, 2005. (Best paper award).
9. M. Held. Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Comput. Geom. Theory Appl.*, 18:95–123, 2001.
10. M.I. Karavelas. A robust and efficient implementation for the segment Voronoi diagram. In *Proc. 1st Int. Symp. Voronoi Diagrams*, pages 51–62, 2004.
11. M.I. Karavelas and M. Yvinec. Voronoi diagram of convex objects in the plane. In *Proc. 11th Europ. Symp. Algorithms*, LNCS, pages 337–348. Springer, 2003.
12. R. Klein, K. Mehlhorn, and S. Meiser. Randomised incremental construction of abstract Voronoi diagrams. *Comput. Geom.: Theory & Appl.*, 3(3):157–184, 1993.
13. R. Ramamurthy and R. T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries - II: Detailed algorithm description. *J. Comput. Appl. Math.*, 102(2):253–277, 1999.
14. R. Ramamurthy and R.T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries I. Theoretical foundations. *J. Comput. Appl. Math.*, 102(1):119–141, 1999.
15. M. Ramanathan and B. Gurumoorthy. Constructing medial axis transform of planar domains with curved boundaries. *Comp.-Aid. Des.*, 35(7):619–632, 2003.