A Real-time and Exact Implementation of the Predicates for the Voronoi Diagram of Parametric Ellipses

Ioannis Z. Emiris*
National University of Athens
Department of Informatics and Telecoms.

George M. Tzoumas†
National University of Athens
Department of Informatics and Telecoms.

Abstract

We study the Voronoi diagram, under the Euclidean metric, of a set of ellipses, given in parametric representation. We use an efficient incremental algorithm and focus on the required predicates. The paper concentrates on INCIRCLE, which is the hardest predicate: it decides the position of a query ellipse relative to the Voronoi circle of 3 given ellipses. We describe an exact, real-time, and complete implementation for INCIRCLE, combining a certified numeric algorithm with algebraic computation. The numeric part leads to a real-time implementation for non-degenerate inputs. It relies on a geometric preprocessing that guarantees a unique solution in a box of parametric space, where a customized subdivision-based method approximates the Voronoi circle tracing the bisectors. Our subdivision method achieves quadratic convergence by exploiting the geometric characteristics of the problem. To achieve robustness, we develop interval-arithmetic techniques, based on the C++ package ALIAS. We switch to an algebraic approach for handling the degeneracies fast. Based on a different algebraic system to model INCIRCLE, we apply real solving and resultant theory. The latter relies on certain symbolic routines which are efficiently implemented in MAPLE. Our approach readily generalizes to arbitrary conics. The paper concludes with experiments showing that most instances run in less than 0.1 sec, on a 2.6GHz Pentium-4, whereas degenerate cases may take up to 13 sec.

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

Keywords: Voronoi diagram, parametric ellipse, robust predicate, symbolic-numeric computation

1 Introduction

The Voronoi diagram is one of the essential tools in robot motion planning amidst obstacles. In this problem, the computation of a path of the moving object can be transformed to a computation of a path in a Voronoi diagram. Applications, such as navigation among objects, benefit the most from the exact Voronoi diagram of ellipses, since ellipses model different kind of obstacles.

Exact Voronoi diagrams have been studied extensively, however the bulk of the existing work in the plane concerns point or linear sites. [McAllister et al. 1996] computes the diagram of convex polygons,

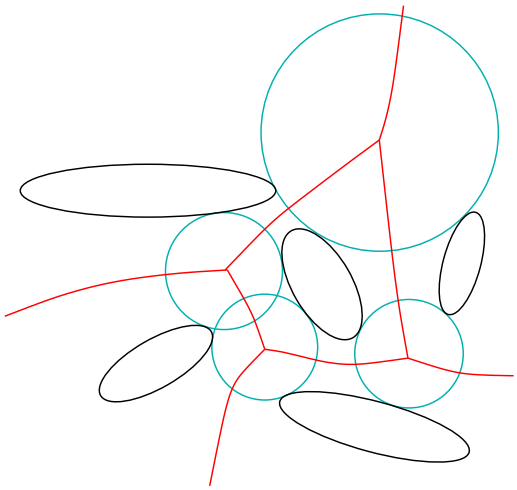

Figure 1: Voronoi diagram of five ellipses

with an approach similar to our subdivision technique, since their algorithm "moves" on the objects' boundary using pruning. Recent efforts have extended Voronoi diagrams to the case where the sites are curves [Alt et al. 2005; Anton 2004] or have non-empty interior [Boissonnat and Karavelas 2003]. In particular, the diagram of circles has been implemented in CGAL [Emiris and Karavelas 2006]; see also the work of [Anton et al. 2002] and [Kim et al. 2001].

The work coming closest to ours towards a complete Voronoi diagram, is [Hanniel et al. 2005]. The authors essentially trace the bisectors in order to compute the Voronoi cells of arbitrary curves up to machine precision. Their algorithm uses floating point arithmetic and their software IRIT [1] works well in practice, with runtimes ranging from a few seconds to a few minutes. Although they argue that their algorithm can be extended to exact arithmetic, it is not clear whether handling the various degenerate configurations may lead to a robustness issue.

Our ambition is to tackle such robustness issues, starting with the Voronoi diagram of ellipses (fig. 1), without compromising efficiency. We focus on non-intersecting ellipses, but our methods are readily applicable to intersecting ellipses, arbitrary conics and, eventually, any planar parametric curve or arc. Our work also extends the state-of-the-art of exact geometric algorithms, as becomes clear from the aforementioned existing work. Notice that exact constructions do not preclude numeric computing as long as the latter certifies its output. This is the premise of this work in order to combine speed and robustness. Hence we refer to our algorithms as symbolic-numeric.

Our approach is based on the incremental algorithm of [Karavelas and Yvinec 2003]. The algorithm is structured according to some basic predicates that must be implemented in the exact computation paradigm. The four predicates are:

*e-mail: emiris@di.uoa.gr
†e-mail: geotz@di.uoa.gr

[1] http://www.cs.technion.ac.il/ irit/

(κ_1) given two ellipses and a point outside of both, decide which is the ellipse closest to the point.

(κ_2) given two ellipses, decide the position of a third one relative to a specific external bitangent of the first two.

(κ_3) given three ellipses, decide the position of a fourth one relative to one (external tritangent) Voronoi circle of the first three; this is the INCIRCLE predicate.

(κ_4) given four ellipses, compute the part of the bisector that changes due to the insertion of a fifth ellipse.

A previous work [Emiris et al. 2006] studied these predicates for the case of *implicit, or algebraic* ellipses, under a purely symbolic approach. Exact implementations of κ_1 and κ_2 were presented in C++ using the algebraic library SYNAPS [2]. The implementations ran in the range of milliseconds, except for INCIRCLE (κ_3) (and, subsequently, κ_4) whose running time was in the range of seconds, a non-satisfactory runtime in a real-time application.

Predicate κ_1 is decided by considering the pencil of a circle and an ellipse. The circle is tangent to the ellipse iff the discriminant of the characteristic polynomial of the pencil vanishes. The smallest real root of the discriminant is the distance between a circle and an ellipse. Therefore to decide the predicate, two algebraic numbers of degree four must be compared. The computed polynomials are derived from the implicit equation of the ellipse. The equations (3) show that conversion from parametric to implicit form is easy. The runtimes of the implementation of this predicate vary from 0.45ms with 10-bit coefficients to 3.68ms with 100-bit coefficients. Predicate κ_2 is handled as follows: Compute the bitangent lines to a pair of ellipses by solving a polynomial of degree four. Then decide the relative position of the query ellipse by computing the sign of a polynomial of degree four at an algebraic point of the same degree. The runtimes vary from 6.15ms with 10-bit coefficients to 73.21ms with 100-bit coefficients. The speed and efficiency of the implementations is due to the small algebraic degree of the involved quantities.

The study of κ_3 showed that there are at most 184 tritangent complex circles to 3 ellipses and this bound is tight. The bound also holds for 3 conics in the plane. Therefore, an algebraic approach will have to do computations with algebraic numbers of degree up to 184. Hence, a purely algebraic approach is not efficient, with today's software. On the other hand, the maximum number of real tritangent circles is an open problem. It has not been possible to find an instance of 3 non-intersecting ellipses with more than 22 real solutions. There is a construction [Ronga 2005] where 3 conics have at least 136 real tritangent circles.

In [Emiris et al. 2006], the authors sketched a subdivision scheme for κ_3 that worked better than generic solvers, although it had linear convergence. It was an improvement compared to exact, numeric and interval-based generic solvers. PHCPACK[3] implements homotopy continuation took about 1 minute to solve systems (6) and (7) in the parametric space, and about 30 sec for an equivalent system in the Cartesian space. With some preliminary experiments [Lazard 2006] Gröbner bases were able to isolate all real roots of the system in about two minutes with GB-Rs[4]. Subdivision-based solvers find all solutions inside an initial box. The iCOs interval-arithmetic solver[5] achieved a running time between one second and four minutes depending on how big the initial domain was. Notice that solving the algebraic system that defines the Voronoi circle is not enough to decide κ_3. One must take into account the query ellipse as well, which is equivalent to solving another similar system.

Predicate κ_4 takes as input ellipses E_t, E_r, E_s, E_h and determines the type of conflict of ellipse E_q with the Voronoi edge whose vertices are the centers of the Voronoi circles defined by E_t, E_r, E_s ($\mathscr{V}_{trs}$) and E_t, E_r, E_h ($\mathscr{V}_{trh}$) respectively. The conflict region is the set of points V on the Voronoi edge where an Apollonius circle of E_t, E_r centered at V intersects E_q, and it may fall into one of six cases [Emiris and Karavelas 2006; Karavelas and Yvinec 2003]: NOCONFLICT, INTERIOR, 123-VERTEX, 124-VERTEX, TWOVERTICES, ENTIREEDGE. To decide κ_4, the authors compute $\mathscr{V}_{trs}$, $\mathscr{V}_{trh}$, $\mathscr{V}_{trq}$ and apply κ_3 on $(\mathscr{V}_{trs}, E_q)$ and $(\mathscr{V}_{trh}, E_q)$, The result of the predicate depends on the results of κ_3 and the order of the tangency points of $\mathscr{V}_{trs}$, $\mathscr{V}_{trh}$, $\mathscr{V}_{trq}$ on E_t.

Recently, we tried another algebraic solver, namely the KRONECKER solver [Giusti et al. 2001; Lecerf 2006], which took about 30 minutes. We also tried ALIAS [6] [Merlet 2000] built-in solvers. The naive subdivision algorithm failed to converge, while gradient based methods converged within a few seconds depending on the initial interval. From these experiments it seems that if we focus inside a box, we are able to quickly approximate a specific solution. Subdivision-based methods indeed work well, as in [Elber and Kim 2001] where the authors find all solutions to the same problem (with circles instead of ellipses) in about 12 sec. Our results manage to improve upon such timings.

In this paper, we focus on the computation of κ_3 which is clearly the most challenging predicate and the only one that is not satisfactorily answered yet; notice that κ_4 is solved by application of κ_3. Our main contribution is a new numeric algorithm that exhibits *quadratic* convergence in order to approximate the Voronoi circle, by exploiting several geometric properties of the problem. Not only our method is faster than generic solvers and other existing implementations, including the one in [Emiris et al. 2006], but also allows us to decide κ_3 before full precision has been achieved. An advantage of this method is that it can be generalized to arbitrary parametric curves and arcs. The algorithm proved to be very efficient, approximating the root with precision up to 10^{-15} in less than 0.1 sec, when using standard floating-point arithmetic. The subdivision methods and more experimental results are discussed in the sequel.

Robustness is guaranteed by interval arithmetic, while exactness is guaranteed by root separation bounds. However, we avoid running the subdivision algorithm until the (theoretical) separation bound is reached. At near-degenerate or degenerate configurations, where floating-point (or even arbitrary) precision is not enough, we switch to an exact algebraic method in the parametric space which corresponds to real solving a univariate polynomial and comparing real algebraic numbers. Our final goal is a CGAL [7] software package for constructing the Voronoi diagram of ellipses.

The paper is organized as follows. The next section discusses representation issues. Section 3 explains how the INCIRCLE is modeled with bitangent and tritangent circles. In section 4 we detail our subdivision method, while 5 shows how we finally decide INCIRCLE and how degenerate cases are handled. Section 6 presents our implementation results along with some implementation details. Finally, section 7 concludes our work and presents some open issues.

[2] http://www-sop.inria.fr/galaad/software/synaps/

[3] http://www.math.uic.edu/∼jan/PHCpack/

[4] http://fgbrs.lip6.fr/salsa/Software/index.php

[5] http://www-sop.inria.fr/coprin/ylebbah/icos/

[6] http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS.html

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

2 Parametric Representation

We assume that an ellipse is given in rational parametric form, i.e. constructively in terms of its rational axes, center and rotation. It is more intuitive for a GUI to offer constructive input rather than require the coefficients of the implicit form. The parametric approach is suitable for tracing the boundary of an ellipse, which is crucial for the subdivision step of the symbolic-numeric algorithm for κ_3. Moreover, given the parametric form it is always possible to derive the implicit one using only rational arithmetic. The opposite is not always possible.

The parametric representation of an ellipse is

$$\begin{aligned} x(t) &= x_c + \alpha\left(\frac{1-w^2}{1+w^2}\right)\left(\frac{1-t^2}{1+t^2}\right) - \beta\left(\frac{2w}{1+w^2}\right)\left(\frac{2t}{1+t^2}\right) \\ &= x_c + \frac{-\alpha(1-w^2)t^2 - 4\beta wt + \alpha(1-w^2)}{(1+w^2)(1+t^2)}, \\ y(t) &= y_c + \alpha\left(\frac{2w}{1+w^2}\right)\left(\frac{1-t^2}{1+t^2}\right) + \beta\left(\frac{1-w^2}{1+w^2}\right)\left(\frac{2t}{1+t^2}\right) \\ &= y_c + 2\frac{-\alpha wt^2 + \beta(1-w^2)t + \alpha w}{(1+w^2)(1+t^2)}, \end{aligned} \tag{1}$$

where $2\alpha, 2\beta$ are the lengths of the major and minor axes, respectively, $t = \tan(\theta/2) \in (-\infty, \infty)$, θ is the angle that traces the ellipse, $w = \tan(\omega/2)$, ω is the rotation angle between the major and horizontal axes and (x_c, y_c) is its center. This representation leaves out of the boundary a single point, called the *i*-point.

The symmetric ellipse (with respect to its center) is

$$\overline{x}(t) = -x(-t) + 2x_c \text{ and } \overline{y}(t) = -y(t) + 2y_c.$$

We call it the *twin* ellipse. Every point of an ellipse is different from its twin point, including the *i*-point. We denote by E_t an ellipse parametrized by t and by $\overline{E_t}$ its twin ellipse. We use twin ellipses in order to deal with intervals that contain *i*-points.

An ellipse has the following implicit equation:

$$E(x,y) := ax^2 + 2bxy + cy^2 + 2dx + 2ey + f \in \mathbb{Q}[x,y]. \tag{2}$$

The coefficients of (2) are expressed by polynomials in the coefficients of (1):

$$\begin{aligned} \chi &= y_c w^2 + 2x_c w - y_c, \\ \psi &= x_c w^2 - 2y_c w - x_c, \\ (1+w^2)^2 a &= 4w^2\alpha^2 + (w-1)^2(w+1)^2\beta^2, \\ (1+w^2)^2 b &= 2(\alpha-\beta)(\alpha+\beta)w(w-1)(w+1), \\ (1+w^2)^2 c &= 4w^2\beta^2 + (w-1)^2(w+1)^2\alpha^2, \\ (1+w^2)^2 d &= -2w\chi\alpha^2 - (w-1)(w+1)\psi\beta^2, \\ (1+w^2)^2 e &= +2w\psi\beta^2 - (w-1)(w+1)\chi\alpha^2, \\ (1+w^2)^2 f &= \chi^2\alpha^2 + \psi^2\beta^2 - (1+w^2)^2\alpha^2\beta^2. \end{aligned} \tag{3}$$

Note that χ and ψ express the equations of the major and minor axes, respectively, evaluated at (x_c, y_c). When an ellipse is given in parametric form constructively (rational axes, center and w), the equations in (3) allow us to transform it to its implicit form.

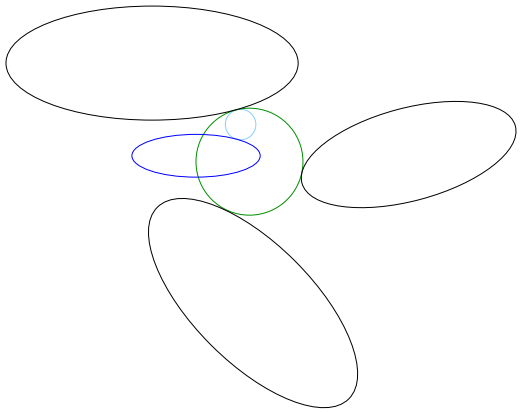

Figure 2: The query ellipse intersects the Voronoi disk

3 The InCircle predicate (κ_3)

Given ellipses E_t, E_r, E_s we want to determine the relative position of ellipse E_h with respect to the Voronoi circle of the first three. The resultt of κ_3 is whether E_h is outside, intersects the open Voronoi disk, or is externally tangent to the Voronoi circle of E_t, E_r, E_s. An example where the query ellipse intersects the Voronoi disk is shown in fig. 2.

In order to decide κ_3, we need some information about the Voronoi circle of three ellipses. The following subsections show how we represent the Voronoi circle as the condition where two bitangent circles coincide.

3.1 The bitangent circle

The *bisector* of two ellipses is the locus of points at equal distance from the two ellipses. Given ellipses E_t, E_r and points P, Q on each of them, the *bisector* is obtained as the intersection V of the normal lines at the ellipses, at P, Q, when $|\overrightarrow{PV}| = |\overrightarrow{QV}|$. This expresses all points on the bisector except for a finite number of them, namely where the two normals are parallel.

The implicit equation of a normal at point t of a parametric curve $(x(t), y(t))$ is:

$$x'(t)x + y'(t)y - x'(t)x(t) - y'(t)y(t) = 0 \tag{4}$$

Point $V(v_1(t,r), v_2(t,r))$ is the solution of a linear system of two equations of the form (4), expressing the normals respectively at points with parameter values t and r. A point defined by parameter value t will also be referred to as point t, or $t \in E_t$. The coordinates' denominator D_{tr} vanishes iff the normals are parallel to each other.

The bisector is

$$\begin{aligned} B(t,r) \quad = \quad & \big(v_1(t,r) - x(t)\big)^2 + \big(v_2(t,r) - y(t)\big)^2 - \\ & \big(v_1(t,r) - x(r)\big)^2 - \big(v_2(t,r) - y(r)\big)^2, \end{aligned} \tag{5}$$

which is rational in t, r with denominator D_{tr}. The numerator is a bivariate polynomial of degree 6 in t and 6 in r. It can be shown that it also vanishes when D_{tr} vanishes. Therefore it includes both bisector points at infinity as well as points where the normal vectors of the two ellipses coincide (i.e. at the minimum distance between two ellipses). For more information on parametric bisectors see [Elbert and Kim 1998]. We now consider the bitangent circles.

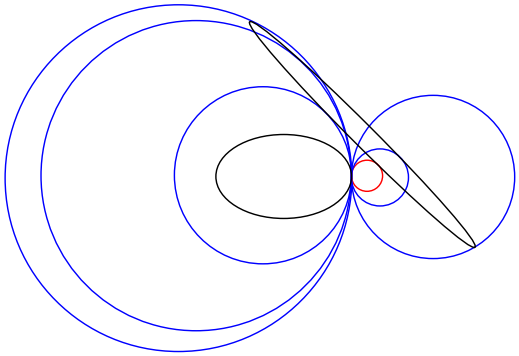

Figure 3: The six bitangent circles: The Apollonius circle is the 3^{rd} from the right

Lemma 3.1 *Given two ellipses and a point on the first, there may exist up to six real bitangent circles, tangent at the specific point. This bound is tight in the worst case.*

Proof. If we fix t, equation (5) has six complex solutions with respect to r. Therefore, six is an upper bound for the number of possible real bitangent circles. Moreover, a configuration of two ellipses that have six real bitangent circles can be attained, see fig. 3. □

Note that only one such circle is *external* to both ellipses. We call this unique external bitangent circle the *Apollonius* circle of the two ellipses, e.g. the third circle from the right in fig. 3. We denote the Apollonius circle of E_t and E_r tangent at points $\hat{t}$ and $\hat{r}$ respectively by $\mathscr{A}_{tr}(\hat{t},\hat{r})$. Since $\hat{r}$ depends on $\hat{t}$, we may omit the latter and write only $\mathscr{A}_{tr}(\hat{t})$.

Given two non-intersecting ellipses E_t, E_r as in fig. 5, the tangency points of any Apollonius circle lie inside their Convex Hull (CH). Thus, for the parametrization (1), there is at least an i-point of E_t, E_r, E_t, E_r that does not lie inside CH. This implies that we can always search for a Voronoi circle within a *continuous* range on the boundary of an ellipse or its twin.

Now, consider all bitangent circles to E_t, E_r, tangent at point t on E_t. Also, consider the lines from t tangent to E_r at points r_1, r_2. They define two arcs on E_r. Arc (r_1, r_2), whose interior points lie on the same side of line $r_1 r_2$ as t, is called a *visible arc*.

Lemma 3.2 *Visible arc (r_1, r_2) contains only tangency points of bitangent circles at t, which are externally tangent to E_r. These points include the tangency point of $\mathscr{A}_{tr}(t)$ on E_r.*

Proof. From a point Q inside the visible arc (fig. 4), an internally tangent circle to E_r cannot be tangent at t, because the tangent line at Q leaves t and E_r on different hyperplanes. External bitangent circles at r_1 and r_2 are tangent to E_t at points t_1, t_2 respectively. Since t lies between t_1 and t_2, there exists some point r between r_1 and r_2 that defines $\mathscr{A}_{tr}(t,r)$. □

The visible arc may also include some other bitangent circles *internally* tangent to E_t. The subset of the visible arc that contains only one tangency point, namely the one corresponding to the Apollonius circle is called an *Apollonius arc*.

Lemma 3.3 *Given is a point $P = (x(t), y(t))$ on E_t. Consider the line l, tangent at P (see fig. 4). If l does not intersect E_r, then the visible arc is an Apollonius arc. Otherwise, the endpoints of the Apollonius arc are: the intersection of l with E_r and the endpoint of the visible arc which lies on the opposite side of E_t with respect to l.*

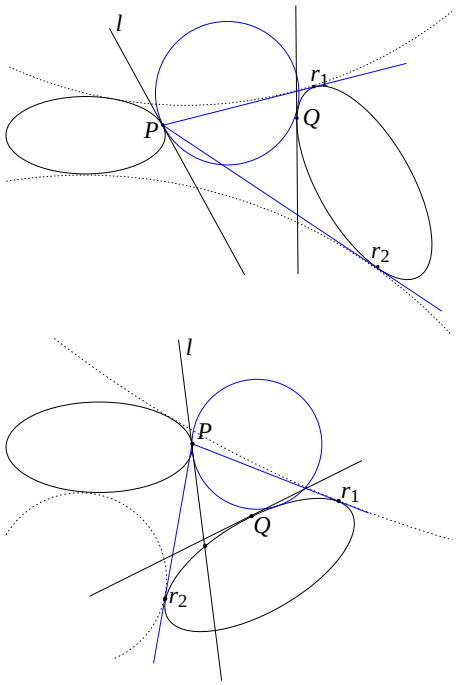

Figure 4: The two cases for defining an Apollonius arc

Proof. If l does not intersect E_r, then it leaves each ellipse in a different hyperplane. In this case, a circle internally tangent to E_t at t, cannot be tangent to E_r as well. Thus, according to the visibility lemma, the visible arc is an Apollonius arc. If l intersects E_r, then a circle internally tangent to E_t at t can be tangent to E_r at a point that lies in the same hyperplane of l as E_t. Therefore, only the part of the visible arc of E_r that lies in the opposite hyperplane is an Apollonius arc. □

We thus obtain arc (r_1, r_2) or $(r_2, \infty) \cup (-\infty, r_1)$ on E_r which contains *only* the tangency point of the Apollonius circle, *isolating* it from the tangency points of non-external bitangent circles.

Corollary 3.4 *Given a point t_0 on E_t, it is possible to determine the unique root r_i of $B(t_0, r)$, from equation (5), which lies on the Apollonius arc of E_r with respect to t_0.*

Corollary 3.5 *The Apollonius arc is a subset of the arc that lies inside the CH.*

3.2 The tritangent circle

In the parametric space, the intersection of two bisectors involves three variables, so in order to express the Voronoi circle, we need the intersection of three bisectors, namely the system:

$$B(t,r) = B(t,s) = B(r,s) = 0. \tag{6}$$

As an alternative, we can consider the system:

$$Q(t,r,s) = B(t,r) = B(t,s) = 0. \tag{7}$$

Here, Q is the condition that makes the three normals of each ellipse intersect at a single point. Q is a polynomial of total degree 12, four in each variable t, r, s. This system has a mixed volume of 432, which bounds the number of common affine roots, see [Cox et al. 2005]. To find the roots, we use resultants, which express

the solvability of polynomial systems. Resultant matrices are such that their determinant is a nontrivial multiple of the resultant itself. In particular, we construct a resultant matrix whose determinant is factored. Let res_s denote the resultant with respect to s. Then we compute two Sylvester resultants:

$$\begin{aligned} R_1(t,r) &= \mathrm{res}_s(Q(t,r,s),B(t,s)) \\ &= \underbrace{(at^{28}r^{24}+\ldots)}_{R_1'}(b_{12}t^{12}+\ldots+b_1t+b_0)(1+t^2)^4 \\ R_2(t) &= \mathrm{res}_r(R_1'(t,r),B(t,r)) \qquad (8) \\ &= (a_{184}t^{184}+\ldots+a_1t+a_0)(c_{12}t^{12}+\ldots+c_1t+c_0)^6\cdot \\ &\quad (1+t^2)^{28}. \end{aligned}$$

We have proven that there are factors which have no real roots, or their roots correspond to the normals lying on the same line. In every example we have tried, if we eliminate all these factors at appropriate powers, we obtain a polynomial of optimal degree (184) that contains all relevant roots. We conjecture that this is the general case. We have proven that the factors exist, but we have no proof for the exponents.

4 Voronoi circle using subdivision

In this section, we present a subdivision scheme to approximate the solution of system (6). Experiments have shown that generic Gradient-based methods run faster for (6) than for (7), most probably because in the former, only two variables are involved in each equation. Our algorithm is a simpler method that exhibits quadratic convergence and exploits the system's geometric symmetry.

Lemma 4.1 *Consider $\mathscr{A}_{tr}(t,r)$, which means that $B(t,r)=0$. If we slide this circle along the boundary of the ellipses, while varying its radius and keeping it tangent to t' and r' respectively, then $t' > t \Longrightarrow r' < r$.*

Proof. The proof is geometrically intuitive or by contradiction. □

The basic idea of our algorithm is the following: Let $(\hat{t},\hat{r},\hat{s})$ be the solution of (6) we are looking for. Now consider the following system:

$$\begin{aligned} B(t_1,r_2)=B(r_2,s_1)=B(s_1,t_2) &= 0 \qquad (9) \\ B(t_2,r_1)=B(r_1,s_2)=B(s_2,t_1) &= 0 \qquad (10) \end{aligned}$$

These two systems look like (6). 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, leads to a construction such that in fig. 6. That is, if we start from a point that is not a tangency point of the Voronoi circle, we compute two bitangent circles for each ellipse. If we are lucky enough, all bitangent circles coincide with the Voronoi circle ($t_1=\hat{t}=t_2$). Otherwise, we have found an interval $[t_1,t_2]$ that contains $\hat{t}$. Now that we have an enclosing interval $[t_1,t_2]$ for $\hat{t}$, we can refine it by choosing a new point t_1' inside this interval and computing a new interval $[t_1',t_2']$ (suppose without any harm that $t_1' < t_2'$). As a consequence of lemma 4.1, 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. 6). Therefore $t_1' \in [t_1,t_2] \Longrightarrow \hat{t} \in [t_1',t_2'] \subset [t_1,t_2]$. 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 temporarily maintain a large number of boxes that may contain a solution and later reject them.

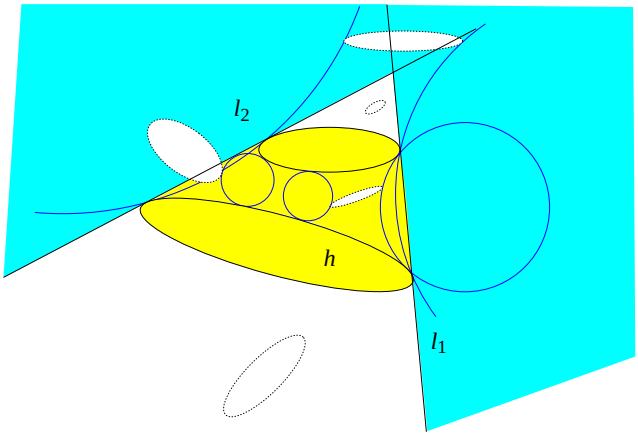

Figure 5: The cases on the number of Voronoi circles, depending on the position of the third dotted ellipse

4.1 Starting interval

First, we show that it is possible to find a box that contains a unique solution. Consider the complements l_1,l_2 of the supporting hyperplanes of the two external bitangents of two ellipses E_1, E_2 and a query ellipse E that does not intersect the other two. Let $|l_i|=0$ or 1 depending on whether $E\cap l_i=\emptyset$ or not. Let C be the interior of the convex hull of E_1,E_2. Then, the number of Voronoi circles is

$$\begin{cases} |l_1|+|l_2|, & \text{if } E\cap C=\emptyset \\ 2-|l_1|-|l_2|, & \text{otherwise} \end{cases}$$

We can find a starting interval that contains the tangency point of the Voronoi circle by computing the external bitangents of each pair of ellipses and taking the intersection of the interior of their convex hulls. After computing the initial interval, we can pick any random point point t_1 to start our algorithm.

It might be necessary to normalize our initial intervals so that they contain the solutions of (9) and (10), shown in fig. 6. We start from value t_1 and solve equations (9) and (10) in the order they appear. At a given step, we have computed a value for parameter x, say $\hat{x}$ and then compute the Apollonius circle at point $\hat{y}$ of the next ellipse, that is we compute $\mathscr{A}_{xy}(\hat{x},\hat{y})$. If $\hat{y}\in[y_1,y_2]$, then we update the corresponding endpoint of the box with $\hat{y}$. The corresponding endpoint depends on how $\hat{y}$ moves, as $\hat{x}$ moves from x_1 to x_2. If $\hat{y}\notin[y_1,y_2]$, then we move backwards, updating $\hat{x}$ by computing the Apollonius circle tangent to the corresponding endpoint of the $[y_1,y_2]$. The normalization procedure in pseudo-code is as follows:

Normalization procedure

INPUT: Initial intervals $[t_1,t_2],[r_1,r_2],[s_1,s_2]$ that contain $(\hat{t},\hat{r},\hat{s})$.

OUTPUT: Small enough intervals $[t_1,t_2],[r_1,r_2],[s_1,s_2]$, so that they contain the solutions of (9) and (10) (fig. 6).

- $X := [t_1,r_2,s_1,t_2,r_1,s_2]$ is a list with cyclic indexing (modulo 6)
- $E := [t,r,s,t,r,s]$
- **for** $i := 1$ **to** 5 **do**
  - find $\mathscr{A}_{E[i-1]E[i]}(X[i],z)$, that is the Apollonius circle of $E[i-1]$ and $E[i]$ at points $X[i]$ and z respectively

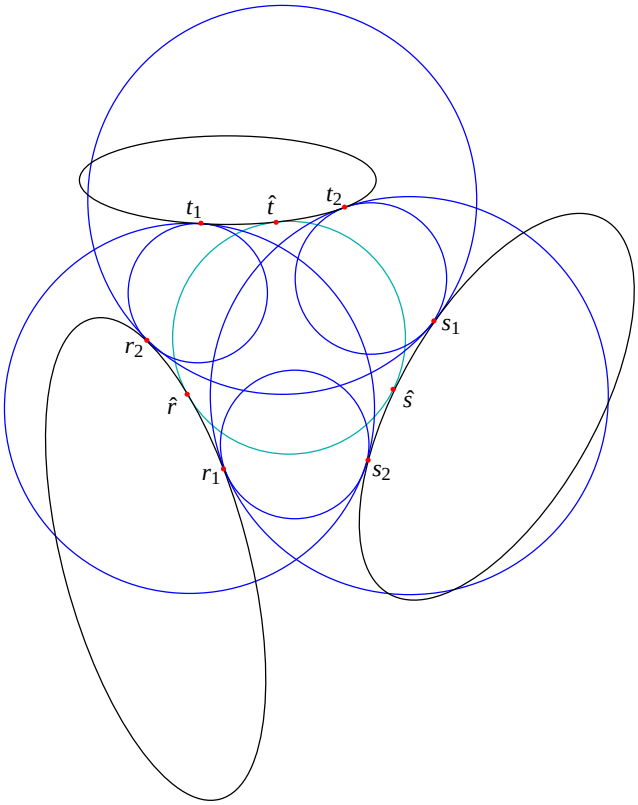

Figure 6: All-pair bitangent circles

– **if** z between $X[i]$ and $X[i+3]$ **then** $X[i] := z$

  else for $j := i$ **downto** 1 **do**

  * find $\mathscr{A}_{E[j]E[j-1]}(X[j], z)$
  * $X[j-1] := z$

There is also the case where two Voronoi circles exist. In this case we assume that we know in advance which one we want and therefore we pick a proper sub-interval.

4.2 Subdivision

Eliminating r_2, s_1 from (9) yields $R(t_1,t_2)$ as a bivariate polynomial equation. Similarly, eliminating r_1, s_2 from (10) yields $R(t_2,t_1)$ as a bivariate polynomial equation. Now we observe that $R(t_1,t_2) = R(t_2,t_1)$, since they have been derived from the same equations with the same coefficients. When $t_1 \neq t_2$, systems (9) and (10) express a perturbation of the Voronoi circle into bitangent circles of each pair of ellipses, as shown in figure 6.

Looking at $R(t_1,t_2)$ we see that t_2 is an implicit function of t_1, say f, that is $f(t_1) = t_2$. Given value t_1, $f(t_1)$ is the value of t_2 after solving (9), shown in fig. 7. Obviously $f(\hat{t}) = \hat{t}$, since we are on a tritangent circle. From lemma 4.1 it follows that $t_1 < t' < \hat{t} \Longrightarrow f(t_1) > f(t') > f(\hat{t})$. That is, as t_1 approaches $\hat{t}$ from the left, t_2 approaches $\hat{t}$ from the right and $\hat{t} \in [t_1,t_2]$.

From the Implicit Function Theorem we have that $\frac{df}{dt_1}$ exists and (by chain rule):

$$\frac{df}{dt_1} = f'(t_1) = -\frac{\partial R/\partial t_1}{\partial R/\partial t_2}$$

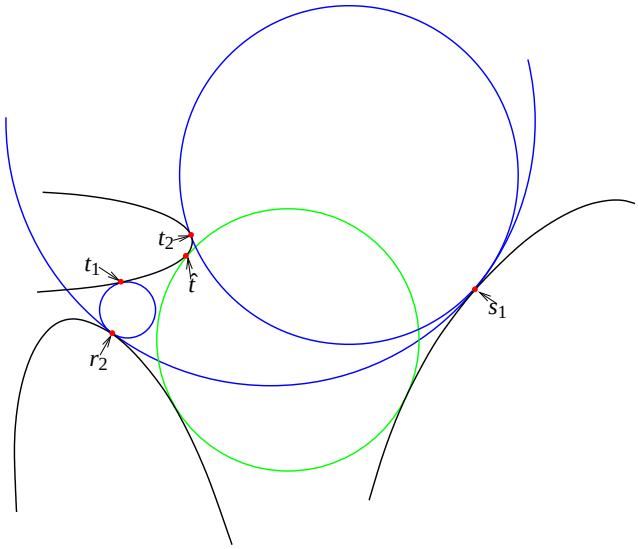

Figure 7: Computing t_2 from t_1 encloses $\hat{t}$

Since $R(t_1,t_2) = R(t_2,t_1)$ we have

$$\left.\frac{\partial R(t_1,t_2)}{\partial t_1}\right|_{t_1=t_2=\hat{t}} = \left.\frac{\partial R(t_1,t_2)}{\partial t_2}\right|_{t_1=t_2=\hat{t}},$$

therefore

$$f'(\hat{t}) = -\left.\frac{\partial R/\partial t_1}{\partial R/\partial t_2}\right|_{t_1=t_2=\hat{t}} = -1.$$

The subdivision algorithm in pseudo-code is as follows:

Subdivision Algorithm

INPUT: Initial intervals $[t_1,t_2],[r_1,r_2],[s_1,s_2]$ that contain unique $(\hat{t},\hat{r},\hat{s})$ and $\sigma \in R$, $\sigma > 0$.

OUTPUT: Subintervals $[t_1,t_2],[r_1,r_2],[s_1,s_2]$ of the given ones, which contain $(\hat{t},\hat{r},\hat{s})$ and $t_2 - t_1 < \sigma$.

1. Start from point t_1 on the first ellipse and solve system (9), with the additional constraint that r_2, s_1, t_2 correspond to Apollonius circles. After computing t_2, also solve (10) in order to obtain intervals for $[r_1,r_2]$ and $[s_1,s_2]$ respectively. Adjust endpoints of each interval, so that the left endpoint is smaller than the right, swapping if necessary.
2. If $t_2 - t_1 < \sigma$ then stop.
3. Set $t_1 := \frac{t_1+t_2}{2}$. Note that the midpoint of $[t_1,t_2]$ could be on the left or on the right of $\hat{t}$. Go to step 1.

Theorem 4.2 (Convergence) *The above subdivision algorithm converges quadratically.*

Proof. For the proof, t_1, t_2 are not generic variables, but have specific values instead, as is the case during the execution of the algorithm. Let

$$\varepsilon = |f(t_1) - t_1|$$

be the error at one iteration of the method. At the next iteration, the new error is $\varepsilon' = |f(\frac{t_1+f(t_1)}{2}) - \frac{t_1+f(t_1)}{2}|$ or equivalently:

$$\varepsilon' = |f(t_1 + \frac{f(t_1)-t_1}{2}) - t_1 - \frac{f(t_1)-t_1}{2}|.$$

Applying Taylor expansion of $f(t_1+\frac{f(t_1)-t_1}{2})$ around point t_1 yields $\varepsilon' = |f(t_1)+\frac{f(t_1)-t_1}{2}f'(t_1)+\frac{(f(t_1)-t_1)^2}{8}f''(\xi)-t_1-\frac{f(t_1)-t_1}{2}|$, with ξ between t_1 and $\frac{t_1+t_2}{2}$. Notice that, from the theory of Taylor expansion, the use of ξ allows us to omit less significant terms. Now ε' becomes:

$$\varepsilon' = \left|\frac{f(t_1)-t_1}{2}(1+f'(t_1))+\frac{\varepsilon^2}{8}f''(\xi)\right|.$$

This time we apply Taylor expansion of $f'(t_1)$ around point $\hat{t}$:

$$f'(t_1) = f'(\hat{t})+(t_1-\hat{t})f''(\xi') = -1+(t_1-\hat{t})f''(\xi'),$$

with ξ' between t_1 and $\hat{t}$, such that it allows us to omit less significant terms. Now remember that $\hat{t}\in[t_1,t_2]$ which means that

$$|t_1-\hat{t}| \le |t_2-t_1| = |f(t_1)-t_1|.$$

Therefore:

$$\varepsilon' = \left|\frac{f(t_1)-t_1}{2}(t_1-\hat{t})f''(\xi')+\frac{\varepsilon^2}{8}f''(\xi)\right| \Longrightarrow$$

$$\varepsilon' \le \frac{\varepsilon^2}{2}|f''(\xi')|+\frac{\varepsilon^2}{8}|f''(\xi)|.$$

Given that $f''(t)$ is a continuous function it takes a minimum and maximum value inside $[t_1,t_2]$, therefore $|f''(\xi)|$ and $|f''(\xi')|$ are bounded by a positive constant C. Finally

$$\varepsilon' \le \frac{5C}{8}\varepsilon^2. \qquad \square$$

5 Deciding κ_3

This section shows how we can decide κ_3. Let $(\hat{t},\hat{r},\hat{s})$ be the solution of (6) which corresponds to the Voronoi circle of E_t,E_r,E_s. Let $\rho_{xy}(\hat{x})$ be the radius of the $\mathscr{A}_{xy}(\hat{x})$. To decide the predicate, we consider the radius $\rho_{th}(\hat{t})$.

Property 5.1 *If $\rho_{th}(\hat{t}) < \rho_{tr}(\hat{t})$ then the query ellipse intersects the Voronoi disk of E_t,E_r,E_h. If $\rho_{th}(\hat{t}) > \rho_{tr}(\hat{t})$ then the query ellipse lies outside the Voronoi circle of E_t,E_r,E_h. If $\rho_{th}(\hat{t}) = \rho_{tr}(\hat{t})$ then the query ellipse is externally tangent to the Voronoi circle.*

Proof. Let $C_q = \mathscr{A}_{th}(\hat{t})$ and $C_V = \mathscr{A}_{tr}(\hat{t})$. By definition, C_V is the Voronoi circle. Let $\hat{h}$ be the tangency point of C_q on ellipse E_h. If $\rho_{th}(\hat{t}) < \rho_{tr}(\hat{t})$, then $\hat{h}$ lies inside C_V, therefore E_h intersects the Voronoi disk C_V. If $\rho_{th}(\hat{t}) > \rho_{tr}(\hat{t})$, then $\hat{h}$ lies outside C_V, therefore E_h lies outside the Voronoi circle C_V. Consequently, if $\rho_{th}(\hat{t}) = \rho_{tr}(\hat{t})$, then $\hat{h}$ lies on the boundary of C_V, therefore E_h is externally tangent to the Voronoi circle C_V. $\square$

Consider the difference of the radii of the above $\mathscr{A}_{th}(\hat{t})$ and $\mathscr{A}_{tr}(\hat{t})$. These circles share one tangency point. Therefore, to compare their radii, it suffices to compare the x-coordinates of their centers (provided that the tangency point and the center do not define a vertical line), denoted by $V_x(\mathscr{A}_{th})$ and $V_x(\mathscr{A}_{tr})$ respectively (see fig. 8). The center of $\mathscr{A}$ is defined as the solution of a system of two linear equations of the form (4). Therefore,

$$V_x(\mathscr{A}_{th}) - V_x(\mathscr{A}_{tr}) = -\frac{\frac{dy(t)}{dt}Q_{thr}(t,h,r)}{D_{th}(t,h)D_{tr}(t,r)},$$

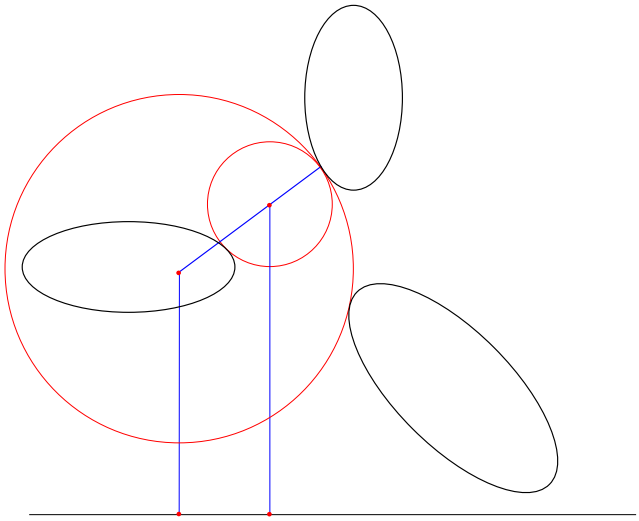

Figure 8: Comparing the radii of two Apollonius circles

where Q is defined in (7), and D corresponds to the determinant that appears as denominator, when solving the linear system. If $dy(t)/dt|_{t=\hat{t}} > 0$, then $\rho_{th}(\hat{t}) < \rho_{tr}(\hat{t}) \Longrightarrow V_x(\mathscr{A}_{th}) < V_x(\mathscr{A}_{tr})$. Similarly, if $dy(t)/dt|_{t=\hat{t}} < 0$, then $\rho_{th}(\hat{t}) < \rho_{tr}(\hat{t}) \Longrightarrow V_x(\mathscr{A}_{th}) > V_x(\mathscr{A}_{tr})$. Therefore, in order to compare the radii it suffices to compute the sign of the following quantity, which is equivalent to computing the sign of each factor separately:

$$\mathscr{S} = -Q_{thr}(t,h,r)\cdot D_{th}(t,h)\cdot D_{tr}(t,r).$$

Consequently, κ_3 can be decided as follows:

sign($\mathscr{S}$)	the query ellipse…
-	intersects the Voronoi disk
+	lies outside the Voronoi circle
0	is externally tangent to the Voronoi circle

5.1 Degeneracies

If the query ellipse is tangent to the Voronoi circle, then the previous sign evaluations will fail for arbitrary small intervals, since Q_{thr} evaluates to zero at the solution of the system. The same also happens with near-degenerate configurations, when our run-time precision is not enough. For near-degenerate configurations, we could always run the algorithm with arbitrary-high precision. This however doesn't eliminate the problem, since there will always be near-degenerate configurations where our precision is not enough to handle. If we do computations with dynamic precision, then we may end up doing operations with very big bitsizes which slows down the overall algorithm considerably. If we are not interested in *exact* computation, we could just report all these configurations as degenerate. But in applications where exact computation is vital, more work has to be done.

At a degenerate configuration, given that we are able to do computations with arbitrary high precision (changing dynamically) the algorithm does not terminate. To overcome this obstacle, we have to use the separation bound theory, which roughly states that given a specific input bitsize, degenerate and non-degenerate configurations are separated by a (very small) constant. Therefore, we could run the algorithm and now that it will terminate when we reach this constant which is precomputed in advance. The problem is that it might need quite many iterations to achieve this constant and we have to

do computations with numbers of huge bitsize. An analysis on the required number of iterations for a subdivision-based algorithm to decide κ_3 can be found in [Emiris et al. 2006]. It states that if the input coefficients have τ bits, then and the order of convergence of our method is $\phi > 1$, then the number of iterations needed for κ_3 is $\log_\phi(1508 + 181944\tau)$ and that for 10-bit input coefficients, a linear convergence scheme will need almost 2 million iterations. On the other hand, quadratic convergence requirees about 20 iterations. Note that these theoretic bounds are quite pessimistic.

As an alternative to the separation bound theory, we can analyse the problem using resultants. Given three ellipses E_t, E_r, E_s and a query one E_h, we compute the resultants of system (7), i.e. $R(t)$ and $R'(t)$, with respect to the two triplets t,r,s and t,r,h, respectively. If all four ellipses share a common Voronoi circle, then $R(t)$ and $R'(t)$ have a common root. In this case $G(t) = \gcd(R(t), R'(t)) \neq 1$. Given $R(t), R'(t)$ and $G(t)$, we can isolate the real roots of $R(t)$, $R'(t)$ and $G(t)$, and obtain the actual separation bound. Now, we may run the subdivision scheme up to this bound, which in all practical cases, is a lot larger than the theoretical one. This way we avoid operations with numbers of large bitsize.

In practice we run the subdivision scheme for a prescribed number of steps, in order to decide all easy (non-degenerate) cases. If we do not get the answer then we switch to the algebraic approach that is presented above. The real roots of $R(t)$, $R'(t)$ and $G(t)$ are represented by isolating intervals. If R and R' do not have common real roots, i.e $G(t) = 1$, then we continue the subdivision algorithm until it halts, since we know that the subdivision process stops. If they have common real roots, i.e $G(t) \neq 1$, then we refine the subdivision interval either until it contains only one real root of $G(t)$ and no other root of R and R', which means that we are in a degenerate configuration, or until it contains no real root of $G(t)$, which means that the subdivision process stops and answers the predicate.

Notice that in all the cases the subdivision scheme runs, in the worst case, up to the real separation bound induced by the real root isolation of $R(t)$ and $R'(t)$.

6 Implementation

The implementation of the numeric part is in C++ using the interval-arithmetic library ALIAS. At each iteration we have to solve the bisector polynomial (5). Fortunately ALIAS provides a fast univariate polynomial solver for this task. Computing the diameter of the approximating intervals during each iteration of the algorithm verified its quadratic convergence. For instance, a diameter sequence was like: [3.01679, 0.978522, 0.303665, 0.0381727, 0.000628676, 1.70776e-07, 1.17684e-14]. Our experiments show that about 8 iterations are enough to approximate the roots of the system with a precision of 10^{-15} in about 80 msec on a Pentium-4 2.6GHz. When deciding κ_3, we have to do some extra computations with the query ellipse. But in this case, we may be able to decide the predicate at a lower precision. Thus, the average running time is again about 80 msec.

The algebraic part is efficiently implemented in MAPLE. We compute the resultant of (7) for the two triplets of ellipses as described in sec. 5.1. The resultant computation is performed by computing two Sylvester resultants, as shown in (9). For 10-bit input coefficients, the resultant computation takes about 6 sec, therefore, for both triplets we spend around 12 sec. Then we isolate the roots of the resultant polynomials in less than a second using the MAPLE package GB-Rs (see footnote 4). Other solvers could also be used here. The total running time is about 13 sec. Since we perform exact arithmetic, the input bitsize is an important factor on the running time of the symbolic algorithm.

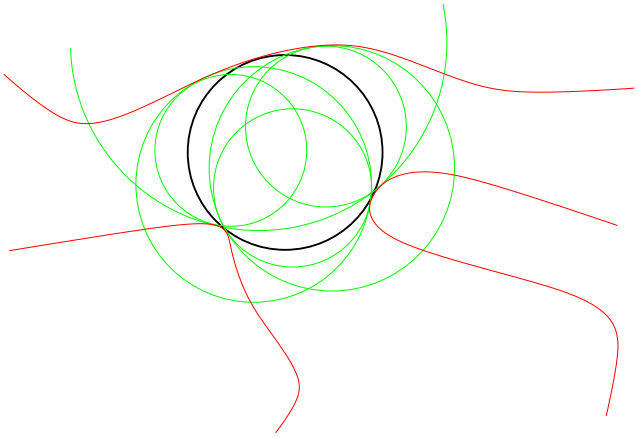

Figure 9: Voronoi circle of three arbitrary curves

7 Conclusion

We presented an efficient algorithm to compute the Voronoi circle of three ellipses and decide predicate κ_3. The numeric part of the algorithm exhibits quadratic convergence and is easy to implement. It can be readily generalized in order to compute the Voronoi circle of arbitrary parametric curves, as shown in fig. 9. The algebraic part allows us to handle degenerate cases by using resultants and real solving. In a complete C++ implementation, we will have to develop efficient symbolic routines to compute the resultant of system (7). In this direction, we plan to develop C++ code using the SYNAPS library that will finally lead to a complete CGAL package.

Acknowledgments George Tzoumas is partially supported by State Scholarship Foundation of Greece, Grant No. 4631. Both authors acknowledge partial support by IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes) and by PYTHAGORAS, project 70/3/7392 under the EPEAEK program of the Greek Ministry of Educational Affairs and EU. The project is co-funded by the European Social Fund & National Resources - EPEAEK II - PYTHAGORAS.

References

ALT, H., CHEONG, O., AND VIGNERON, A. 2005. The Voronoi Diagram of Curved Objects. *Discrete and Computational Geometry 34*, 3 (Sep), 439–453.

ANTON, F., BOISSONNAT, J.-D., MIOC, D., AND YVINEC, M. 2002. An exact predicate for the optimal construction of the additively weighted Voronoi diagram. In *Europ. Workshop Comput. Geom.*

ANTON, F. 2004. *Voronoi diagrams of semi-algebraic sets*. PhD thesis, The University of British Columbia.

BOISSONNAT, J.-D., AND KARAVELAS, M. 2003. On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres. In *Proc. SODA*, 305–312.

COX, D., LITTLE, J., AND O'SHEA, D. 2005. *Using Algebraic Geometry*, 2nd ed. No. 185 in GTM. Springer, New York.

ELBER, G., AND KIM, M.-S. 2001. Geometric constraint solver using multivariate rational spline functions. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, ACM Press, New York, NY, USA, 1–10.

ELBERT, G., AND KIM, M.-S. 1998. Bisector curves of planar rational curves. *Computer-Aided Design 30*, 1089–1096.

EMIRIS, I., AND KARAVELAS, M. 2006. 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.

EMIRIS, I. Z., TSIGARIDAS, E. P., AND TZOUMAS, G. M. 2006. The predicates for the voronoi diagram of ellipses. In *Proc. ACM 22th Annual Symposium on Computational Ceometry (SoCG)*, ACM Press, New York, NY, USA, 227–236.

GIUSTI, M., LECERF, G., AND SALVY, B. 2001. A Gröbner free alternative for polynomial system solving. *J. Complex. 17*, 1, 154–211.

HANNIEL, I., MUTHUGANAPATHY, R., ELBER, G., AND KIM, M.-S. 2005. Precise Voronoi cell extraction of free-form rational planar closed curves. In *Proc. 2005 ACM Symp. Solid and phys. modeling*, 51–59. (Best paper award).

KARAVELAS, M., AND YVINEC, M. 2003. Voronoi diagram of convex objects in the plane. In *Proc. ESA*, 337–348.

KIM, D.-S., KIM, D., AND SUGIHARA, K. 2001. Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry. *CAGD 18*, 563–585.

LAZARD, D., 2006. Personal communication.

LECERF, G., 2006. Personal communication.

MCALLISTER, M., KIRKPATRICK, D., AND SNOEYINK, J. 1996. A compact piecewise-linear Voronoi diagram for convex sites in the plane. *Discrete Comput. Geom. 15*, 73–105.

MERLET, J. P. 2000. ALIAS: an interval analysis based library for solving and analyzing system of equations. In *SEA*.

RONGA, F., 2005. Personal communication.