# Low Degree Algorithms for Computing and Checking Gabriel Graphs

Giuseppe Liotta

Department of Computer Science
Brown University
Providence, Rhode Island 02912

**CS-96-28**
October 1996

# Low Degree Algorithms for Computing and Checking Gabriel Graphs *
# (Extended Abstract)

*Giuseppe Liotta*
Center for Geometric Computing
Department of Computer Science, Brown University
115 Waterman Street, Providence, RI 02912–1910, USA
gl@cs.brown.edu

**Abstract**

In the context of robust computational geometry we focus on the problem of computing and checking Gabriel graphs with algorithms that are not affected by degenaracies and have low arithmetic demand. A simple and practical linear-time algorithm is presented that constructs the Gabriel Graph of a finite point set on the plane from its Delaunay diagram. The degree of the algorithm, i.e. a measure of the arithmetic precision required to carry out exact computations, is evaluated and proved to be optimal. The problem of certifying the correctness of an algorithm that computes the Gabriel graph is also investigated and an otpimal degree procedure is proposed.

*Keywords:* Computational geometry, robustness, Gabriel graph, Delaunay diagram.

*Research supported in part by the National Science Foundation under grant CCR–9423847, by the U.S. Army Research Office under grants DAAH04–93–0134 and DAAH04–96–1–0013, and by the N.A.T.O.-C.N.R. Advanced Fellowships Programme.

# 1 Introduction and Overview

In many problems of pattern recognition and computational morphology, such as clustering and computational approaches to perception, one is given a set of points on the plane and is asked to display the underlying *shape* of the set by constructing a graph whose vertices are the points and whose edges are segments connecting pairs of points. In clustering it is desired to have an algorithm that connects two points that belong to the same cluster. In computational perception we would like an algorithm that joins pairs of points so that the resulting graph is perceptually meaningful in some sense (for more details on these problems see also [25]).

Several graphs that capture the notion of shape of a set $S$ of points on the plane have been described in the literature. In the survey by Toussaint [15] such graphs are classified by using the notion of *proximity* between sets of points. In a proximity graph points are connected by edges if and only if they are deemed *close* by some proximity measure. It is the measure that determines the type of graphs that result. Minimum spanning trees [22], Gabriel graphs [11], relative neighborhood graphs [26], Delaunay diagrams (often referred to as Delaunay triangulations if all internal faces are three-cycles) [22] , all describe different notions of closeness between the points of the input point set. By exploiting the inclusion relationship between these global descriptors of closeness (the Delaunay diagram is a supergraph of the Gabriel graph that is a supergraph of the relative neighborhood graph that is a supergraph of the minimum spanning tree), the shape of a set of points can be analyzed by considering the spectrum of progressively less detailed descriptions when going from the Delaunay diagram to the minimum spanning tree (see, e.g. [16, 23]). Clearly, fast and reliable algorithms to construct the wanted proximity subgraphs of the Delaunay diagram play a crucial role within this approach.

The problem of efficiently computing subgraphs of the Delaunay diagram of a set of points is the subject of a rich body of literature in computational geometry. One of the first papers is due to Cheriton and Tarjan [6], who show a $O(n)$-time algorithm to compute an Euclidean minimum spanning tree from a Delaunay diagram with $n$ vertices. The Gabriel graph can be computed from the Delaunay diagram in $O(n)$-time by using the algorithm of Matula and Sokal [19]. The first $O(n \log n)$-time algorithm to compute a relative neighborhood graph from a Delaunay diagram is due to Supowit [24], who improves the previous $O(n^2)$-time bound showed by Toussaint [26]. Jaromczyk and Kowaluk [13] show that the relative neighborhood graph can be computed from the Delaunay diagram in $O(n\alpha(n,n))$-time, where $\alpha(\cdot)$ is the inverse of the Ackermann's function. The bound is reduced to $O(n)$ by Jaromczyk, Kowaluk, and Yao [14]. A different proof of this last result is given by Lingas [17]. For a complete survey on algorithms that compute proximity drawings see [15].

Existing algorithms, however, often accomplish the asymptotically optimal efficiency at the expenses of simplicity; also they often rely on simplifying assumptions on the input configuration (no three points are collinear and no four points are co-circular) that make them fail when implemented in practice. For example, the linear-time algorithm of [14] is based on complex UNION-FIND data structures and assumes that all internal faces of the Delaunay diagram are three-cycles.

On the other hand, issues of robustness, experimentation, and implementation have become central to the development of geometric algorithms. Simplifying assumptions ruling out degenerate configurations of the input and an excessive attention to the asymptotic performance have in fact considerably limited the desired technology transfer from computational geometry to applied fields like robotics, computer graphics, pattern recognition, and GIS. To overcome such discomfort, a considerable effort is being invested in re-visiting classical computational geometry problems and in re-designing simple, effective and reliable solutions for such problems. A limited list of papers that are devoted to robust computational geometry includes [20, 1, 5, 4, 7, 10, 8].

In this paper, we re-visit the problem of efficiently computing and checking the Gabriel graph of a set of points $S$. In the context of methodologies intended to confer robustness to geometric computation, we analyze the efficiency of the proposed algorithms within a realistic model, in which the arithmetic precision that is needed to carry out exact computation is relevant. To this aim, we adopt the concept of *degree* of an algorithm [18], which characterizes, up to a small additive constant, the number of bits needed by a geometric algorithm to be error free, i.e. to construct topologically consistent outputs even in the presence of degenerate configurations of the input.

The main contributions of this paper are listed below.

- We present a lower bound to the degree of the problem of computing the Gabriel graph of a set of points from its Delaunay diagram. The technique to achieve the lower bound consists in analyzing the complexity of the geometric predicates that must be evaluated by any algorithms that correctly solve the problem. We show that the geometric tests executed by such algorithms are multivariate polynomials of degree 2. This means that if the input points are represented by pairs of $b$-bit integers, a correct implementation of the primitives that execute the geometric tests require to handle integers of at most $2b + O(1)$, where the constant $O(1)$ depends on the number of terms in the polynomial to be evaluated.
- We study the efficiency of existing algorithms that compute the Gabriel graph of a set of points from its Delaunay diagram. We compare a brute-force $O(n^2)$-time approach to the well-known $O(n \log n)$-time algorithm by Matula and Sokal [19] that constructs the Gabriel graph from the Delaunay diagram in two steps. In the first step the Voronoi diagram is computed from and the coordinates of its vertices are stored with exact arithmetic, i.e. as rational numbers (pairs of integers). In the second step all edges in the Delaunay diagram that do not intersect the corresponding dual edge are deleted. The resulting graph is the Gabriel graph. We show the existence of a sharp trade off between computational efficiency and degree required to carry out exact computation. Namely, while the brute-force method has optimal degree 2, the algorithm by Matula and Sokal has degree 6.
- We present a new characterization of Gabriel graphs in terms of Delaunay diagrams. We show two different applications of our characterization. The first application reconciles robustness with efficiency by providing a linear-time, optimal-degree algorithm that computes the Gabriel graph of a set of points on the plane from its Delaunay diagram. The algorithm is robust and easy to implement. Also, it is fast in practice, requiring at most two geometric tests to determine whether an edge of the Delaunay diagram belongs to the underlying Gabriel graph.
- As a second application of our characterization, we study a geometric program checking problem. Namely, Let $\mathcal{P}$ be a program whose input is a finite set of distinct points on the plane and whose output $\mathcal{O}(S)$ is claimed to be the Gabriel graph of $S$. We present a *checker* for $\mathcal{P}$, i.e. a program that analyzes $\mathcal{O}(S)$ and either certifies its correctness or returns a whitness that $\mathcal{O}(S)$ is not a Gabriel graph. We present an optimal-degree linear-time checker. We show that our procedure satisfies the requirements of correctness, simplicity, and efficiency invocated by Mehlhorn et al. [20] as mandatory for effective geometric checkers.

Furthermore, as a side effect of the lower bounds techniques presented in this paper, a lower bound on the degree of the segment intersection problem is established.

# 2 Preliminaries

We give first some basic geometric definitions and notation. We then briefly describe the notion of *degree* of a geometric algorithm. For more details the reader is referred to [22] and to [18].

## 2.1 Voronoi Diagrams, Gabriel Graphs, and Delaunay Diagrams

Let $S$ be a finite set of $n$ distinct points on the plane; we assume $n \geq 2$. The locus of points of the plane that are closer to a point $p_i \in S$ than to any other $p_j \in S$ is the *Voronoi polygon* of $p_i$ and is denoted by $V(i)$. $V(i)$ is a convex polygonal region (possibly unbounded) given by the intersection of $n-1$ half-planes and having at most $n-1$ sides. The *Voronoi diagram* of $S$ is the planar subdivision defined by the union of the $n$ Voronoi polygons $V(i)$ associated to each $p_i \in S$. Figure 1 (a) shows a set $S$ and its Voronoi diagram. The vertices and deges of $V(S)$ are called *Voronoi vertices* and *Voronoi edges*, respectively.

The *Delaunay diagram* of $S$, denoted by $D(S)$, is the straight-line dual of $V(S)$, i.e. the planar subdivision obtained by drawing a straight-line segment between each pairs of points of $S$ whose Voronoi polygons share an edge. If $S$ does not contain *degeneracies*, i.e. no four points $S$ are co-circular and no three points of $S$ are collinear, then all internal faces of $D(S)$ are three-cycles. A Delaunay diagram such that all internal faces are three-cycles is often referred to as a *Delaunay triangulation*. The vertices and edges of $D(S)$ are called *Delaunay vertices* and *Delaunay edges*, respectively. Given an edge $(u, v) \in D(S)$, we will often need to refer to the edge of $V(S)$ shared by the Voronoi polygons $V(u)$ and $V(v)$; we call such edge the *dual edge* of $(u, v)$. An example of a Delaunay diagram is shown in Figure 1 (b), where the Delaunay edges are solid lines and the underlying Voronoi edges are dotted lines. Notice that a Delaunay edge and its dual edge may or may not intersect. If they intersect, the intersection point is the midpoint of the Delaunay edge [21].

The *Gabriel graph* [11] of $S$, denoted by $GG(S)$, is the planar subdivision obtained with the following rule: Two vertices $u, v \in S$ are connected by a straight-line segment if and only if

$$d^2(u, v) < d^2(u, w) + d^2(v, w), \text{ for all } w \in S, w \neq u, v.$$

An equivalent definition of $GG(S)$ is that $u$ and $v$ are connected by a straight-line segment if and only if the disk $D[u, v]$ having $u$ and $v$ as antipodal points is *empty*, i.e. $D[u, v]$ does not contain any other point of $S - \{u, v\}$. $D[u, v]$ is also called the *Gabriel disk* of $u$ and $v$ and it is assumed to be a closed set. An example of a Gabriel graph is depicted in Figure 1, where also the Gabriel disk of an edge $(u, v)$ is represented. Matula and Sokal [20] prove that the Gabriel graph is a subgraph of the Delaunay diagram, i.e. every straight-line segment representing an edge of $GG(S)$ is also an edge of $D(S)$. In the same paper, an elegant characterization of those edges of $D(S)$ that belong to $GG(S)$ is also given. To present such characterization we need a preliminary definition: Two straight-line segments are said to *properly intersect* if they share a point that is not one of their endpoints.

**Theorem 1** *[19] Let $e$ be and edge of $D(S)$ and let $e_v$ be the dual edge of $e$. Edge $e$ belongs to $GG(S)$ if and only if $e$ and $e_v$ properly intersect.*

## 2.2 Degree of Geometric Algorithms

In the next sections we will be measuring the efficiency of geometric algorithms within a finer framework than the "big-Oh" analysis. To this aim we adopt the notion of *degree* that was first introduced in [18] and independently in [3] as a measure of the arithmetic precision needed to carry out exact computation during the execution of geometric algorithms. For reasons of space we give here only the basic definitions. The reader is referred to [18] for a more detailed description of the concepts in this section.

A geometric algorithm executes computations of two types: *tests* (also called *predicates*) and *constructions*. Tests are associated with branching decisions in the algorithm that determine the flow of control, constructions produce the output data of the algorithm. While approximations in the execution of constructions are acceptable as long as their maximum absolute error does not exceed the resolution required by the application (such as spacing of raster lines in graphics), approximations in the execution of tests may produce an incorrect branching of the algorithm and give rise to *structurally* incorrect results. Therefore, tests are much more critical, and their execution must be carried out with complete accuracy. The *degree* of a geometric algorithm characterizes the complexity of the test computations that the algorithm executes.

We consider algorithms that evaluate multivariate polynomials over their variables. A *primitive variable* is an input variable of the algorithm and has conventional arithmetic degree 1. Input variables are reasonably assumed to be expressed with $b$ bits, for some small integer $b$. The arithmetic degree of a variable $v$ is the arithmetic degree of the multivariate polynomial $E$ that computes $v$. The arithmetic degree of $E$ is the maximum (in the homogeneous case, the common) arithmetic degree of its monomials. The arithmetic degree of a monomial is the sum of the arithmetic degrees of its variables.

We say that an algorithm has *degree* $d$ if its tests involve the evaluation of multivariate polynomials of arithmetic degree $d$. An immediate consequence of the definition is that if an algorithm has degree $d$ and its

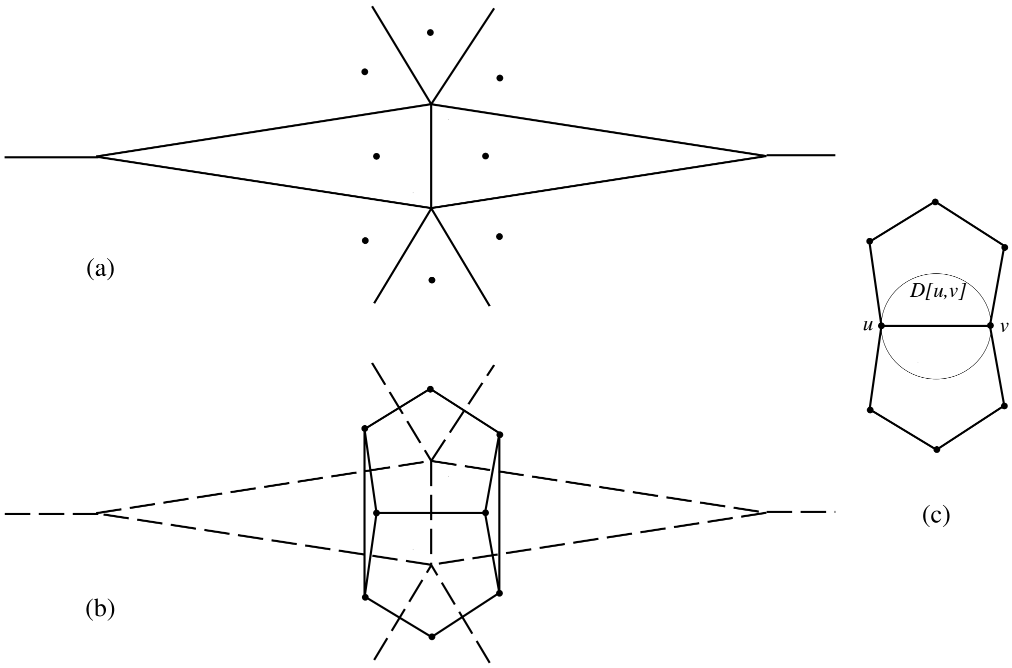


Figure 1: (a) A Voronoi Diagram. (b) A Delaunay diagram. (c) A Gabriel graph.

input variables are $b$-bit integers, then all the test computations can be carried out with at most $db + O(1)$ bits. This justifies our use of the degree of an algorithm to characterize the precision required to execute error-free tests.

We say that a problem $\Pi$ has *degree* $d$ if any algorithm that solves $\Pi$ has degree at least $d$. In [18] it is proven that the degree of the *nearest neighbor problem* is 2. An immediate implication of this resultis the following.

**Corollary 1** *Let $u$, $v$, and $w$ be three points of the plane. The problem of determining whether $w$ is closer to $u$ or to $w$ has degree* 2.

# 3 The Degree of Computing Gabriel Graphs

We first study the degree of the problem of computing a Gabriel graph from a Delaunay diagram. Then we analyze the performance of existing algorithms that solve such problem and show a sharp trade off between their asymptotic behavior and the arithmetic precision needed to correctly evaluate the predicates.

## 3.1 Lower bound

The study of the lower bound is based on showing that the computation of a Gabriel graph from a Delaunay diagram has the same degree as the problem of correctly testing whether two segments properly intersect. In fact, for Theorem 1, the problem of computing a Gabriel graph is equivalent to testing for each edge $(u, v)$ of the Delaunay diagram, whether $(u, v)$ properly intersects its dual Voronoi edge.

The next two lemmas are devoted to define a lower bound on the degree of the segment intersection problem. Let $w$, be a point of the plane and let $s$ be a straight-line segment with endpoints $u$ and $v$. Assuming that $s$ is oriented from $u$ to $v$, the which-side-test(w,s) consists of asking whether $w$ lies to the left, to the right, or on segment $s$.

**Lemma 1** *The* which-side-test(w,s) *can be executed on primitive variables with optimal degree 2.*

**Sketch of proof:** Let $u \equiv (x_u, y_u)$, $v \equiv (x_v, y_v)$, and $w \equiv (x_w, y_w)$. It is well known (see, e.g. [22]) that executing the which-side-test(w,s) is equivalent to determining the sign of the following determinant.

$$\Delta = \begin{vmatrix} x_w & y_w & 1 \\ x_u & y_u & 1 \\ x_v & y_v & 1 \end{vmatrix} = x_u y_v - x_v y_u - x_w y_v + x_v y_w + x_w y_u - x_u y_w .$$

Notice that $\Delta$ is a multivariate polynomial of degree 2 if all variables are assumed to be primitive. It remains to prove that degree 2 is optimal. Suppose that one could execute the which-side-test (w,s) with a degree 1 procedure, then he/she would also have a degree 1 procedure to determine the sign of determinant $\Delta$. We show below that $\Delta$ cannot be expressed as the product of two degree 1 polynomials of the variables $x_w, y_w, x_u, y_u, x_v, y_v$. Which implies that executing the test with degree 2 is optimal.

Suppose that there existed constants $a', a'', b', b'', c', c'', d', d'', e', e'', f', f''$ such that:

$$\begin{aligned} & x_u y_v - x_v y_u - x_w y_v + x_v y_w + x_w y_u - x_u y_w = \\ & (a' x_u + b' y_u + c' x_v + d' y_v + e' x_w + f' y_w) \cdot (a'' x_u + b'' y_u + c'' x_v + d'' y_v + e'' x_w + f'' y_w). \end{aligned}$$

The above equality implies $a'a'' = 0$, since there cannot be a term $a'a''x_u^2$. However, either $a'$ or $a''$ is not 0 because of nonzero terms having $x_u$ as a factor. Assume, w.l.o.g. that $a' \neq 0$. This implies that $b'' = c'' = 0$, since there are not terms of the type $x_u y_u$ and of the type $x_u x_v$. As a consequence, the constant $b'$, and $c'$ are not 0, since there are terms having as a factor $y_u$ and $x_v$, respectively. However, because of the term $x_v y_u$, we should have either $b'c'' \neq 0$ or $c'b'' \neq 0$, a contradiction. □

The above lemma assumes that the entries of the determinant are primitive variables, each of degree 1; since $\Delta$ is an homogeneous multivariate polynomial, the overall degree is the sum of the degrees of the two primitive variables involved in each of the terms of $\Delta$. Clearly, the degree becomes higher if the terms of $\Delta$ contain non-primitive variables. We will present later an example of a which-side-test that requires the evaluation of a degree 6 multivariate polynomial.

Lemma 1, allows us to prove the following.

**Lemma 2** *Testing whether two segments on the plane properly intersect has degree 2.*

**Sketch of proof:** Let $s'$ and $s''$ be two segments on the plane and let $u'$, $v'$, $u''$, and $v''$ be the endpoints of $s'$ and $s''$, respectively. Segments $s'$ and $s''$ properly intersect if and only if the following two conditions are verified: (i) $u'$ and $v'$ are on opposite sides of $s''$ and (ii) $u''$, and $v''$ are on opposite sides of $s'$. Thus if one had a degree 1 procedure to test whether $s'$ and $s''$ properly intersect, he/she would have a degree 1 procedure to execute the following four tests: which-side-test (u',s"), which-side-test (v',s"), which-side-test (u",s'), and which-side-test (v",s'), contradicting Lemma 1. □

Lemma 2 and Theorem 1 give rise to the main result of this section.

**Theorem 2** *The problem of computing the Gabriel graph of a set of points on the plane from its Delaunay diagram has degree 2.*

In the next section we prove that the lower bound established by Theorem 2 is tight. To this aim we introduce a new geometric test primitive, and show that by using it, an optimal degree algorithm can be designed. We also show that computing the Gabriel graph by using the which-side-test instead, gives rise to a high degree algorithm.

## 3.2 The which-side-test vs. the in-Gabriel-disk-test

An optimal-degree algorithm can be readily designed by making use of the following geometric primitive. Given a segment $s$ and a point $p$ on the plane, the in-Gabriel-disk-test (p,s) verifies whether $v$ is in the disk having the endpoints of $s$ as antipodal points.

**Lemma 3** *The* in-Gabriel-disk-test (p,s) *can be executed on primitive variables with optimal degree* 2.

**Sketch of proof:** We show first that the test can be executed with degree 2, and then that such degree is optimal. Let $c$ be the center of the disk $D[u,v]$. Vertex $p$ is outside $D[u,v]$ if and only if $d(p,c)-d(u,c)>0$, that in turn is equivalent to verifying that $d^2(p,c)-d^2(u,c)>0$. Observe that $d^2(p,c)-d^2(u,c)$ is a degree 2 polynomial if all variables involved are primitive variables. The fact that there cannot be a degree 1 procedure that executes the test descends from Corollary 1. □

Let $D(S)$ be the Delaunay diagram of $S$. The *brute-Force algorithm* computes $GG(S)$ by verifying, for every edge $(u,v)\in D(S)$, whether the Gabriel disk of $u$ and $v$ is empty and if so, by adding $(u,v)$ to the set of edges of $GG(S)$.

**Lemma 4** *The brute-force algorithm computes the Gabriel graph of a set $S$ of $n$ points on the plane from the Delaunay diagram of $S$ in $O(n^2)$ time with optimal degree* 2.

**Sketch of proof:** The asymptotic bound on the time complexity immediately follows from the fact that the Delaunay diagram $D(S)$ is planar and that $n-2$ tests are executed for each edge of $D(S)$ to verify whether it is also an edge of the Gabriel graph $GG(S)$. Each test corresponds to the execution of a in-Gabriel-disk-test. Since all variables involved in such test are primitive, it follows from Lemma 3 that the degree of the algorithm is 2. Because of Theorem 2, degree 2 is optimal. □

A linear-time algorithm to compute the Gabriel graph from the Delaunay diagram is based on Theorem 1 and was first suggested by Matula and Sokal [19]. We refer to such algorithm as the *conventional algorithm*, since it is usually accepted by computational geometers as "the algorithm" to compute a Gabriel graph (see, for example [22, 21, 9]). The conventional algorithm consists of two steps. In the first step the Voronoi diagram $V(S)$ of $S$ is computed from $D(S)$ and the coordinates of its vertices are stored with exact arithmetic, i.e. as rational numbers (pairs of integers). In the second step, for every edge $e\in D(S)$ it is tested whether $e$ properly intersects its corresponding dual edge $e_v\in V(S)$. Unfortunately, as the next lemma shows, the conventional approach is characterized by a very high degree.

**Lemma 5** *The conventional algorithm computes the Gabriel graph of a set $S$ of $n$ points on the plane from the Delaunay diagram of $S$ in $O(n)$ time with degree* 6.

**Sketch of proof:** Let $e=(u,v)$ be a Delaunay edge and let $e_v=(p,q)$ be the corresponding dual edge in the Voronoi diagram. By the reasoning in the proof of of Lemma 2, determining whether $e$ and $e_v$ properly intersect is equivalent to executing the following four tests: which-side-test (u,$e_v$), which-side-test (v,$e_v$), which-side-test (p,$e$), and which-side-test (q,$e$).

Consider the which-side-test (u,$e_v$). We show first that such test implies determining the sign of a degree 6 multivariate polynomial, and then we show that such polynomial cannot be expressed as the product of lower degree polynomials.

Let $u\equiv(x_u,y_u)$, let $p\equiv(x_p,y_p)$ be equidistant from three points of $S$ $a\equiv(x_a,y_a)$, $b\equiv(x_b,y_b)$, $c\equiv(x_c,y_c)$, and let $q\equiv(x_q,y_q)$ be equidistant from three points $b\equiv(x_b,y_b)$, $c\equiv(x_c,y_c)$, and $d\equiv(x_d,y_d)$. Answering the which-side-test (u,$e_v$) is equivalent to determining the sign of the following determinant.

$$\Delta=\begin{vmatrix} x_u & y_u & 1\\ x_p & y_p & 1\\ x_q & y_q & 1\end{vmatrix}=\begin{vmatrix} x_u & y_u & 1\\ \frac{X_p}{2W_p} & \frac{Y_p}{2W_p} & 1\\ \frac{X_q}{2W_q} & \frac{Y_q}{2W_q} & 1\end{vmatrix}=\frac{1}{2W_pW_q}\begin{vmatrix} x_u & y_u & 1\\ X_p & Y_p & W_p\\ X_q & Y_q & W_q\end{vmatrix}=\frac{\Delta'}{2W_pW_q},$$

where

$$X_p=\begin{vmatrix} x_a^2+y_a^2 & y_a & 1\\ x_b^2+y_b^2 & y_b & 1\\ x_c^2+y_c^2 & y_c & 1\end{vmatrix},\quad Y_p=\begin{vmatrix} x_a & x_a^2+y_a^2 & 1\\ x_b & x_b^2+y_b^2 & 1\\ x_c & x_c^2+y_c^2 & 1\end{vmatrix},\quad W_p=\begin{vmatrix} x_a & y_a & 1\\ x_b & y_b & 1\\ x_c & y_c & 1\end{vmatrix}$$

and $X_q$, $Y_q$, and $W_q$ have similar expressions obtained replacing in the above determinants $x_c$ with $x_d$ and $y_c$ with $y_d$. The sign of $\Delta$ is studied by evaluating the signs of $W_p$, $W_q$ and of $\Delta'$. It can be easily verified that $\Delta'$ is a degree 6 multivariate polynomial.

□

Lemmas 4 and 5 exhibit a sharp trade off between computational efficiency and degree. Namely, the brute-force approach, based on the in-Gabriel-disk-test, has optimal degree 2 but is rather unefficient in terms of time complexity. On the other hand, the linear-time conventional approach, based on the which-side-test, requires the explicit computation of the Voronoi vertices and this dramatically affects the degree of the algorithm.

Observe that a degree 6 algorithm implies that a $k$-bit arithmetic unit can handle with native precision the computation of the Gabriel graph for points in a grid of size at most $2^{\frac{k}{6}} \times 2^{\frac{k}{6}}$. For example, if $k = 32$, the point that can be treated with native precision belong to a grid of size at most $2^6 \times 2^6$, that is unacceptably small for most of pattern recognition and computational morphology applications.

One of the results of the next section is to close the gap between degree and time complexity shown by Lemmas 4 and 5.

# 4 Computing and Checking with Optimal Degree

In this section we present an alternative characterization of Gabriel graphs in terms of Delaunay diagrams and discuss two different applications of such characterization. A first application reconciles robustness with efficiency by providing a linear-time, optimal-degree algorithm that computes the Gabriel graph of a set of points on the plane from its Delaunay diagram. The second application shows an optimal-degree solution to the problem of checking the correctness of a program that computes Gabriel graphs.

Our characterization generalizes a previous result presented in [2] to solve a graph drawing problem. The main difference between our characterization and the one underlying the conventional algorithm (Theorem 1) is that we are able to detect what edges of the Delaunay diagram belong to the Gabriel graph with a constant number of comparisons, but without looking at any dual edges. This allows us to design linear-time optimal-degree algorithms whose tests are of the type in-Gabriel-disk-test and involve only primitive variables.

**Theorem 3** *Let $D(S)$ be the Delaunay diagram of a finite set of distinct points on the plane and let $(u,v)$ be an edge of $D(S)$. Let $f'$ and $f''$ be the two faces sharing $(u,v)$. $(u,v)$ is an edge of the Gabriel graph $GG(S)$ if and only if one of the following conditions is verified.*

1. *Both $f'$ and $f''$ consists of more than three edges.*
2. *$f'$ has more than three edges, $f''$ is a triangle, and the angle opposite to $(u,v)$ in $f''$ is acute.*
3. *Both $f'$ and $f''$ are triangles and the angles opposite to $(u,v)$ in both $f'$ and $f''$ are acute.*

**Sketch of proof:** We start by proving the first Statement 1. If $f'$ and $f''$ consist of more than three edges, all vertices of $f'$ (of $f''$) are co-circular, by the definition of Delaunay diagrams. This implies that $D[u,v]$ is completely contained in $f' \cup f''$. Hence, $D[u,v]$ is empty, since there cannot be any points in the interior of either $f'$ or $f''$.

We prove now Statement 2. $D[u,v]$ consists of two semi-circles: $D'[u,v]$ is the semi-circle contained inside $f'$, $D''[u,v]$ is $D[u,v] - D'[u,v]$. By the same reasoning as in the proof of Statement 1, $D'[u,v]$ is empty. Let $q$ be the vertex of $f''$ opposite to edge $(u,v)$. If $\angle uqv$ is acute, then $D''[u,v]$ is completely contained in the disk defined by the three vertices $u$, $v$, and $q$. But such disk is empty, since $f''$ is a face of the Delaunay diagram. It follows that also $D'[u,v]$, and hence $D[u,v]$ are empty. Vice-versa, if $(u,v)$ is an edge of the Gabriel graph, then $q \notin D[u,v]$, which implies that $\angle uqv$ is acute.

Finally, the proof of Statement 3 is a variation of the proof of Statement 2. □

## 4.1 Computing with optimal degree

An optimal degree linear-time algorithm that computes the Gabriel Graph from the Delaunay diagram directly descends from Theorem 3. Algorithm Optimal Gabriel has as input the Delaunay diagram of a set $S$ of $n$ distinct points on the plane and produces as output the Gabriel graph of $S$. We make the assumption that the Delaunay diagram $D(S)$ is correct, i.e. we assume that $D(S)$ has been computed by an algorithm that correctly handles degenerate configurations of the input points such as collinearities and co-circularities. For example, one can use the algorithm of Guibas and Stolfi [12]. The Guibas-Stolfi algorithm robustly computes $D(S)$ and stores it in a suitable data structure, called *quad-edge*, that allows to determine in $O(1)$ time for each edge $(u, v) \in D(S)$, what are the two faces sharing $(u, v)$.

In order to emphasize the simplicity of its implementation in practice, we give below a Pascal-like description of Algorithm Optimal Gabriel. We denote with $E(GG)$ the edges of the Delaunay diagram that are also edges of the Gabriel graph. Also, $f'$ and $f''$ denote the two faces sharing edge $(u, v) \in D(S)$.

**Algorithm** Optimal Gabriel
*input:* $D(S)$.
*output:* $GG(S)$.
**begin**
$E(GG) = \emptyset$
**for each** edge $(u, v) \in D(S)$
**do begin**
**if** both $f'$ and $f''$ have more than three vertices,
**then** $E(GG) = E(GG) \cup \{(u, v)\}$
**if** $f'$ has more than three vertices and $f''$ is a triangle $\triangle(uvq)$,
**then if** $\angle uqv < \frac{\pi}{2}$
**then** $E(GG) = E(GG) \cup \{(u, v)\}$
**if** $f'$ is a triangle $\triangle(uvp)$ and $f''$ is a triangle $\triangle(uvq)$,
**then if** $\angle upv < \frac{\pi}{2}$ **and** $\angle uqv < \frac{\pi}{2}$
**then** $E(GG) = E(GG) \cup \{(u, v)\}$
**end**
**end**
**end** Algorithm.

**Theorem 4** *Let $S$ be a finite set of $n$ distinct points on the plane.* Algorithm Optimal Gabriel *computes the Gabriel graph of $S$ from the Delaunay diagram of $S$ in $O(n)$ time and with optimal degree* 2.

**Sketch of proof:** The correctness of Algorithm Optimal Gabriel derives from Theorem 3. The bound on the time complexity derives from the fact that at most two tests are executed for each edge $(u, v) \in D(S)$, in order to determine whether $(u, v)$ also belongs to $E(GG)$. Observe that each of such tests is a in-Gabriel-disk-test executed on primitive variables. The optimality follows from Theorem 2 and from Lemma 3. □

## 4.2 Checking with optimal degree

As a second application of Theorem 3 we study a geometric program checking problem. Namely, we tackle the problem of certifying the correctness of a program that computes a Gabriel graph of a set $S$ of points.

Let $\mathcal{P}$ be a program whose input is a finite set of distinct points on the plane and whose output $\mathcal{O}(S)$ is claimed to be the Gabriel graph of $S$. A *checker* for program $\mathcal{P}$ verifies whether $\mathcal{O}(S)$ is actually the Gabriel graph of $S$. If the check is in the affirmative, than the checker certifies the correctness of $\mathcal{O}(S)$; otherwise, it returns a whitness that $\mathcal{O}(S)$ is not correct, i.e. some evidence that $\mathcal{O}(S)$ cannot be a Gabriel graph.

In what follows we present Algorithm Optimal Check. We assume that $\mathcal{A}$ computes $\mathcal{O}(S)$ by first computing the Delaunay diagram. This is a reasonable assumption since all known efficient algorithms compute Gabriel graphs by first constructing $D(S)$ and then deleting from $D(S)$ those edges that do not belong to $GG(S)$ (see e.g. [15]).

We follow the approach proposed by Mehlhorn et al. [20] to the verification of geometric algorithms. According to such approach, we slightly augment the tasks executed by program $\mathcal{P}$ in order to facilitate the checker. Namely, we ask $\mathcal{P}$ to output with $\mathcal{O}(S)$ also $D(S)$. Observe that this does not affect the asymptotic performance of $\mathcal{P}$.

Algorithm Optimal Check receives as input $\mathcal{O}(S)$ and $D(S)$ and produces as output a *certificate*, that is either a message of correctness, or an edge $(u, v)$ such that $(u, v) \in \mathcal{O}(S)$, but $(u, v) \notin D(S)$, or a pair $< (u, v), p >$ where $(u, v)$ is an edge of $\mathcal{O}(\mathcal{S})$ and $p$ is a point of $S$ such that $p \in D[u, v]$. We assume that $D(S)$ has been correctly computed. An efficient checker for a Delaunay diagram is presented in [20].

**Algorithm** Optimal Check
*input:* $\mathcal{O}(S)$, $D(S)$.
*output:* Certificate.
**begin**
    error= **false**
    **for each** edge $(u, v) \in \mathcal{O}(S)$
    **do begin**
        **if** $(u, v) \in D(S)$
        **then begin**
            **for each** face $f$ such that $(u, v) \in f$ and $f$ is a triangle $\triangle(uvp)$
            **do if** $\angle upv \geq \frac{\pi}{2}$
                **then begin**
                    **print** "Error: point $p$ is in the Gabriel disk of $(u, v)$"
                    error = **true**
                **end**
        **end**
        **else begin**
            **print** "Error: edge $(u, v)$ is not an edge of $D(S)$"
            error = **true**
        **end**
    **end**
    **if not** error **then print** "Program $\mathcal{P}$ computes the Gabriel graph of $S$"
**end**
**end** Algorithm.

**Theorem 5** Algorithm Optimal Check *verifies the correctness of an algorithm that computes the Gabriel graph of a finite set of n distinct points on the plane in $O(n)$ time and with optimal degree 2.*

**Sketch of proof:** The correctness of Algorithm Optimal Check is a consequence Theorem 3. The bound on the time complexity descend from the fact that at most two in-Gabriel-disk-test are executed for each $(u, v) \in \mathcal{O}(S)$. Since each in-Gabriel-disk-test is executed on primitive variables, from Lemma 3 the algorithm has degree 2. If degree 2 were not optimal, then there would exist a degree 1 procedure to detect whether an edge of the Delaunay diagram belongs or not to the Gabriel graph. Which contradicts Theorem 2. □

Melhlhorn et al. [20] define the main requirements for a program checker as correctness, simplicity, and efficiency. Theorem 5 proves the correctness and the efficiency of Algorithm Optimal Check. Although there is not a formal quantifyer for estimating the simplicity of an algorithm, we believe that the easiness of the Pascal-like code, the optimal degree of the strategy, and the low number of geometric tests that are executed on each edge of $\mathcal{O}(S)$ may give some evidence of the simplicity of the proposed approach.

## 5 Open Problems

Several questions remain open. For example,

1. Extend the result of this paper to the robust computation of other proximity subgraphs of the Delaunay diagram. To this respect, a family of particular interest are the so-called *$\beta$-graphs* [16, 23]. $\beta$-graphs are an infinite family of proximity graphs for which the definition of proximity varies according to the value of a parameter $\beta$. For values of $\beta$ such that $1 \leq \beta \leq 2$, $\beta - graphs$ are subgraphs of the Delaunay diagram.
2. Study the degree of the problem of computing the Delaunay diagram of a set of points in the plane. It has been proven that the algorithm by Guibas and Stolfi [12] has degree 4 [18]. It would be very interesting to understand whether a lower degree is achievable.
3. An immediate implication of Lemma 2 is that the problem of reporting the intersections among a finite set of segments on the plane has degree 2. Although the segment intersection problem is a classical subject of study in computational geometry, none of the asymptotically efficient algorithms that solve the problem have optimal degree (see, e.g. [22]). On the other hand, a brute-force quadratic time algorithm is straight-forward. Closing the gap between asymptotic efficiency and optimal degree seems to be a very challenging task.

## Acknowledgments

I am grateful to Franco Preparata, Roberto Tamassia, Luca Vismara, and Paola Vocca for their encouragement, their comments, their friendship.

## References

[1] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. Preparata, and M. Yvinec. Evaluating signs of determinants using single-precision arithmetic. Research Report 2306, INRIA, BP93, 06902 Sophia-Antipolis, France, 1994.

[2] P. Bose, W. Lenhart, and G. Liotta. Characterizing proximity trees. *Algorithmica*, pages 83–110, 1996. (special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia).

[3] C. Burnikel. *Exact Computation of Voronoi Diagrams and Line Segment Intersections.* PhD thesis, Technische Fakultät der Universität des Saarlandes, Saarbrücken Germany, March 1996. Available at URL: http://www.mpi-sb.mpg.de:80/ burnikel/thesis/.

[4] C. Burnikel, K. Mehlhorn, and S. Schirra. How to compute the Voronoi diagram of line segments: Theoretical and experimental results. In *Proc. 2nd Annu. European Sympos. Algorithms (ESA '94)*, volume 855 of *Lecture Notes in Computer Science*, pages 227–239. Springer-Verlag, 1994.

[5] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 16–23, 1994.

[6] D. Cheriton and R. E. Tarjan. Finding minimum spanning trees. *SIAM J. Comput.*, 5:724–742, 1976.

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

[8] D. P. Dobkin. Computational geometry and computer graphics. *Proc. IEEE*, 80(9):1400–1411, Sept. 1992.

[9] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[10] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9:66–104, 1990.

[11] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[12] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4:74–123, 1985.

[13] J. W. Jaromczyk and M. Kowaluk. A note on relative neighborhood graphs. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 233–241, 1987.

[14] J. W. Jaromczyk, M. Kowaluk, and F. Yao. An optimal algorithm for constructing $\beta$-skeletons in the $l_p$ metric. *SIAM journal on computing.* to appear.

[15] J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80(9):1502–1517, Sept. 1992.

[16] D. G. Kirkpatrick and J. D. Radke. A framework for computational morphology. In G. T. Toussaint, editor, *Computational Geometry*, pages 217–248. North-Holland, Amsterdam, Netherlands, 1985.

[17] A. Lingas. A linear-time construction of the relative neighborhood graph from the delaunay triangulation. *Computation Geometry: Theory and Applications*, pages 199–208, 1994.

[18] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries in implicit voronoi diagrams. Technical Report CS-96-16, Center for Geometric coputing, Comput. Sci. Dept., Brown Univ., Providence, RI, 1996.

[19] D. W. Matula and R. R. Sokal. Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane. *Geogr. Anal.*, 12:205–222, 1980.

[20] K. Mehlhorn, S. Näher, T. Schilz, S. Schirra, M. Seel, R. Seidel, and C. Uhrig. Checking geometric programs or verification of geometric structures. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 159–165, 1996.

[21] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, England, 1992.

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

[23] J. D. Radke. On the shape of a set of points. In G. T. Toussaint, editor, *Computational Morphology*, pages 105–136. North-Holland, Amsterdam, Netherlands, 1988.

[24] K. J. Supowit. The relative neighborhood graph with an application to minimum spanning trees. *J. ACM*, 30:428–448, 1983.

[25] G. Toussaint. *Computational Morphology.* North-Holland, 1988.

[26] G. T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recogn.*, 12:261–268, 1980.