Robust Proximity Queries: an Illustration of Degree-driven Algorithm Design[*]

Giuseppe Liotta[†] *Franco P. Preparata*[‡] *Roberto Tamassia*[§]

Abstract In the context of methodologies intended to confer robustness to geometric algorithms, we elaborate on the exact computation paradigm and formalize the notion of degree of a geometric algorithm, as a worst-case quantification of the precision (number of bits) to which arithmetic calculation have to be executed in order to guarantee topological correctness. We also propose a formalism for the expeditious evaluation of algorithmic degree. As an application of this paradigm and an illustration of our general approach, we consider the important classical problem of proximity queries in 2 and 3 dimensions, and develop a new technique for the efficient and robust execution of such queries based on an implicit representation of Voronoi diagrams. Our new technique gives both low degree and fast query time, and for 2D queries is optimal with respect to both cost measures of the paradigm, asymptotic number of operations and arithmetic degree.

1 Introduction

The increasing demand for efficient and reliable geometric software libraries in key applications such as computer graphics, geographic information systems, and computer-aided manufacturing is stimulating a major renovation in the field of computational geometry. The inadequacy of the traditional simplified framework has become apparent, and it is being realized that, in order to achieve an effective technology transfer, new frameworks and models are needed to design and analyze geometric algorithms that are efficient in a practical realm.

The real-RAM model with its implicit infinite-precision requirement, has proved unrealistic and needs to be replaced with a realistic finite-precision model where geometric computations can be carried out either exactly or with a guaranteed error bound. This has motivated a great deal of research on the subject of robust computational geometry (see, e.g., [6, 7, 12, 13, 25, 27]). For an early survey of the different approaches to robust computational geometry the reader is referred to [16].

To a first, rough, approximation, robustness approaches are of two main types: perturbing and nonperturbing. Perturbing approaches transform the given problem into one that is intended not to suffer from well-identified shortcomings; nonperturbing approaches are based on the notion of "exact" (rather than "approximate") computations, with the assumption that (bounded-length) input data are error free. In this category falls the exact geometric computation paradigm independently advocated by Yap [28] and by the Saarbrücken school [3, 4], and so does our approach. Within this paradigm, we introduce the concept of *degree* of a geometric algorithm, which characterizes, up to a small additive constant, the arithmetic precision (i.e., number of bits) needed by a large class of geometric algorithms. Namely, if the coordinates of the input points of a degree-d algorithm within this class are b-bit integers, then the algorithm may be required on some instances to perform arithmetic computations with bit precision $d(b + O(1))$.

Theoretical analysis and experimental results show that multiprecision numerical computations take up most of the CPU time of exact geometric algorithms (see, e.g., [17]). Thus, we believe that, in defining the efficiency of a geometric algorithm, the degree should be considered as important as the asymptotic time complexity and should correspondingly play a major role in the design stage. In fact, the principal thrust of this paper is to present algorithm degree as a major design criterion for geometric computation. Research along these lines involves re-examining the entire rich body of computational geometry as we know it today.

[*]Research supported in part by the National Science Foundation under grant CCR-9423847, by the U.S. Army Research Office under grant DAAH04-96-1-0013, and by the EC ESPRIT Long Term Research Project ALCOM-IT under contract 20244.

[†]Dipartimento di Informatica e Sistemistica, Universitá di Roma "La Sapienza", Via Salaria 113, Roma 00198, Italy. Work by this author performed when he was with the Center for Geometric Computing at Brown University. liotta@dis.uniroma1.it

[‡]Center for Geometric Computing, Department of Computer Science, Brown University, 115 Waterman Street, Providence, RI 02912-1910, USA. franco@cs.brown.edu

[§]Center for Geometric Computing, Department of Computer Science, Brown University, 115 Waterman Street, Providence, RI 02912-1910, USA. rt@cs.brown.edu

In this paper, we consider as a test case a problem area, geometric proximity, which plays a major role in applications and has recently attracted considerable attention because, due to its demands of high precision for exact computation, it is particularly appropriate in assessing effectiveness of robust approaches (see, e.g., [13]). In particular, we shall illustrate the role played by the degree criterion if one wishes to comply with the exact-computation paradigm.

To illustrate the approach, we recall that Voronoi diagrams of 2D point sites are the search structures which permit answering a proximity query without evaluating all query-point/site distances. Therefore, given the set of sites, their Voronoi diagram is computed and supplied as a planar subdivision to a point-location procedure. Assuming that the coordinates of all input data (also called *primitive* points) are b-bit integers, the coordinates of the points computed by the algorithm (referred to here as *derived* points, e.g., the vertices of a Voronoi diagram of points and segments) must be stored with a representation scheme that supports rational or algebraic numbers as data types (through multiprecision integers). Specifically, the coordinates (x, y) of a Voronoi vertex are rational numbers given by the ratio of two determinants (of respective orders 3 and 2) whose entries are integers of well-defined maximum modulus. The fundamental operation used by any point location algorithm is the point-line discrimination, which consists of determining whether the query point q is to the left or to the right of an edge between vertices v_1 and v_2. For the case of the Voronoi diagram $V(S)$, this is equivalent to evaluating the sign of a 3×3 determinant whose rows are the homogeneous coordinates of q, v_1, and v_2, a computation that needs about $6b$ bits of precision. This should be compared with the $O(n)$-time brute-force method that computes the (squares of the) distances from q to all the sites of S, and executes arithmetic computations with only $2b$ bits of precision (which is optimal).

Guided by the low-degree criterion, in this paper we present a technique — complying with the exact-computation paradigm — which uses a new point-location data structure for Voronoi diagrams, such that the test operations executed in the point location procedure are distance comparisons, and are therefore executed with optimal $2b$ bits of precision. Hence, our approach reconciles efficiency with robustness, and advocates an object-oriented programming style where access to the geometry of Voronoi diagrams in point-location queries is encapsulated in a small set of geometric test primitives. It must be pointed out that distance comparisons have already been used nontrivially for proximity search (extremal-search method [11]). However, we shall show that the latter method fails to achieve optimal degree because the search is based on predicates requiring $4b$ bits of precision; moreover, the high overhead of the search technique (which uses the hierarchical polytope representation [8]) casts some doubts on the practicality of the method.

The main results of this work are summarized in Table 1. We show that previous methods for proximity queries exhibit a sharp tradeoff between degree and query time. Namely, low degree is achieved by the slow brute-force search method, while fast algorithms based on point-location in a preprocessed Voronoi diagram or on the extremal-search method have high degree. Our new technique gives instead both low degree and fast query time, and is optimal with respect to both cost measures for queries in sets of 2D point sites.

For proofs and details omitted in this extended abstract, the reader is referred to the full paper [20].

2 Degree of Geometric Problems

The numerical computations of a geometric algorithm are basically of two types: tests (predicates) and constructions. The two types of computations have clearly distinct roles. Tests are associated with branching decisions in the algorithm that determine the flow of control, whereas constructions are needed to produce the output data of the algorithm. Approximations in the execution of constructions are acceptable, since approximate results are perfectly tolerable, provided that the error magnitude does not exceed the resolution required by the application. On the other hand, approximations in the execution of tests may produce an incorrect branching of the algorithm. Such event may have catastrophic consequences, giving rise to *structurally* incorrect results. Te exact-computation paradigm therefore requires that tests be executed with total accuracy. We shall therefore characterize geometric algorithms on the basis of the complexity of their test computations. Any such computation consists of evaluating the sign of an *algebraic expression* over the input variables, constructed using an adequate set of operators, such has $\{+, -, \times, \div, \sqrt[k]{\ }, \ldots\}$. As we shall show below, the expressions under considerations are equivalent to multivariate polynomials.

We make here the reasonable assumption that input data be *dimensionally consistent*, i.e., that if an entity with the physical dimension of a length is represented with b bits, then one with the dimension of an area be represented with $2b$ bits, and so on. Under the hypothesis of dimensional consistency (where point coordinates are b-bit entries), a polynomial expressing a test is homogeneous because all of its monomials must have the same physical dimension. A *primitive variable* is an input variable of the algorithm and has conventional arithmetic degree 1. The arithmetic degree of

Query	Method	Degree	Time
nearest neighbor	brute-force distance comparison	2 *	$O(n)$
	point location in explicit Voronoi diagram	6	$O(\log n)$ *
	extremal-search method	4	$O(\log n)$ *
	point location in implicit Voronoi diagram	2 *	$O(\log n)$ *
k-nearest neighbors and circular range search	brute-force distance comparison	2 *	$O(n)$
	point location in explicit order-k Voronoi diagram	6	$O(\log n + k)$ *
	point location in implicit order-k Voronoi diagram	2 *	$O(\log n + k)$ *
nearest neighbor among points and segments	brute-force distance comparison	6	$O(n)$
	point location in explicit Voronoi diagram	64	$O(\log n)$ *
	point location in implicit Voronoi diagram	6	$O(\log n)$ *
3D nearest neighbor	brute-force distance comparison	2 *	$O(n)$
	point location in explicit 3D Voronoi diagram	8	$O(\log^2 n)$
	point location in implicit 3D Voronoi diagram	3	$O(\log^2 n)$

Table 1: Comparison of the degree and running time of algorithms for some fundamental proximity query problems. A * denotes optimality. The technique introduced in this paper (*point location in an implicit Voronoi diagram*) outperforms previous methods and is optimal for 2D queries.

a polynomial expression E is the common arithmetic degree of its monomials. The arithmetic degree of a monomial is the sum of the arithmetic degrees of its variables.

Definition 1 *An algorithm has* degree d *if its test computations involve the evaluation of multivariate polynomials of arithmetic degree at most d. A problem Π has* degree d *if any algorithm that solves Π has degree at least d.*

Recently, Burnikel [2] has independently defined the notion of *precision of an algorithm*, which is equivalent to our notion of degree of an algorithm. Also, our definition of degree is related to that of *depth of derivation* proposed by Yap [27, 28].

Motivated by a standard feature of geometric algorithms, we also make the assumption that every multivariate polynomial of degree d used in tests depends upon a set of size s (a small constant) of primitive variables. Therefore a multivariate polynomial has $O(s^d)$ distinct monomials with bounded integer coefficients, so that the maximum value of the multivariate polynomial is expressible with at most $db + d \log s$ bits. A consequence of Definition 1 and of the above assumption is the following fact, which justifies our use of the degree of an algorithm to characterize the precision required in test computations.

Lemma 1 *If an algorithm has degree d and its input variables are b-bit integers, then all the test computations can be carried out with $d(b + O(1))$ bits.*

Typically the support of a geometric test is not naturally expressed by a multivariate polynomial, but, rather, by a pair (E_1, E_2) of expressions involving the four arithmetic operations, powering, and the extraction of square roots, and the test consists of comparing the magnitudes of E_1 and E_2. Such expressions always have a physical dimension (a length, an area, a volume, etc.), so that if they have the form of ratios, the degree of the numerator exceeds that of the denominator.

Expressions such as E_1 and E_2 can be viewed as a binary tree, whose leaves represent input variables and whose internal nodes are of two types: binary nodes, which are labeled with an operation from the set $\{+, -, \times, \div\}$, and unary nodes, which are labeled either with a power or with a square root extraction (notice that we restrict ourselves to this type of radicals). If no radical appears in the trees of E_1 and E_2, then the test is trivially equivalent to the evaluation of the sign of a polynomial, since E_i is a rational function of the form $\frac{N_i}{D_i}$ ($i = 1, 2$, N_i, D_i are not both trivial polynomials and $D_i \neq 0$) and $E_1 \geq E_2 \iff (-1)^{\sigma(D_1)+\sigma(D_2)}(N_1 D_2 - N_2 D_1) \geq 0$, where $\sigma(E) = 1$ if $E < 0$ and $\sigma(E) = 0$ if $E \geq 0$. (Note that the above predicate implies the inductive assumption that the signs of lower degree expressions N_1,N_2, D_1, and D_2 are known.) Suppose now that at least one of the trees of E_1 and E_2 contains radicals. We prune the tree so that the pruned tree contains no radicals except at its leaves (notice that pruned subtrees may themselves contain radicals). Then N_i and D_i ($i = 1, 2$) can be viewed as polynomials whose variables are the radicals and whose coefficients are (polynomial) functions of non-radicals. Given a polynomial E in a set of radicals, for any radical R in this set, we can express E as $E = E''R + E'$ where neither E'' nor E' contains R. Then $E \geq 0 \iff E''R \geq -E'$. The resulting expression $({E''}^2 R^2 - {E'}^2)$ does not contain R. Therefore by this device we can eliminate one radical. This shows that by the four rational operations we can reduce the sign test to the computation of the signs of a collection of multivariate polynomials.

We now present a very simple, but useful, formalism that enables us to rapidly evaluate the degree of the multivariate polynomial which uniquely determines the sign of the original algebraic expression.

The terms involved in the formal manipulations are of two types, generic and specific. *Generic* terms have the form α^s (for a formal variable α and an integer s), representing an unspecified multivariate polynomial of degree s over primitive variables. *Specific* terms have the form ρ_j, for some integer index j, representing a specified expression involving the operators $\{+, -, \times, \div, \sqrt{}\}$. The terms are members of a free commutative semiring, i.e., addition and multiplication are associative and commutative, addition distributes over multiplication, and specific terms can be factored out. Beside these conventional algebraic rules, we have a set of *rewriting rules* of the form $A \rightarrow B$, meaning that the sign of A is *uniquely determined* by the sign of B. Note, the two signs not always coincide: indeed the correspondence between the signs of the sides depends upon the evaluated signs of other expressions of lower degree.

We have seven rules, whose correctness can be proved with elementary algebra. The rules are:

$$\begin{array}{lccc}
(1) & \rho_j & \longrightarrow & \alpha^s \\
(2) & \alpha^s\alpha^r & \longrightarrow & \alpha^{s+r} \\
(3) & \alpha^s + \alpha^s & \longrightarrow & \alpha^s \\
(4) & -\alpha^s & \longrightarrow & \alpha^s \\
(5) & \frac{\rho_j}{\rho_i} \pm \frac{\rho_h}{\rho_i} & \longrightarrow & \rho_j \pm \rho_h \\
(6) & \frac{\rho_j}{\rho_i} \pm \frac{\rho_h}{\rho_k} & \longrightarrow & \rho_j\rho_k \pm \rho_i\rho_h \\
(7) & \rho_i \pm \rho_j & \longrightarrow & \rho_i^2 - \rho_j^2.
\end{array}$$

The preceding discussion establishes the following:

Theorem 1 *Rules* (1) – (7) *are adequate to evaluate the degree of a multivariate polynomials which uniquely determine the sign of an arbitrary algebraic expression involving square roots.*

While the above rules represent an adequate formalism for obtaining an upper bound to the degree of an algorithm, more subtle is the corresponding lower-bound question. In other words, given a predicate $\mathcal{P}$ that is essential to the solution of a given problem, what is the inherent degree of $\mathcal{P}$? Suppose that predicate $\mathcal{P}$ is expressed by a polynomial P of degree d, and we must decide whether the value of P is positive, negative, or zero. Can we answer this question by computing a discrete (ternary) function f of analogous evaluations of irreducible polynomials $P_1, \ldots, P_k$ of maximum degree smaller than d? Clearly, f changes value only when some P_j changes sign (exactly, when the value of P_j passes by 0). Thus, a 0 of P corresponds to a 0 of some P_j. Moreover, as the arguments of P_j vary while P_j remains 0, so does f and hence P. Therefore, P vanishes at all points for which P_j vanishes and, for a well-known theorem of polynomial algebra (see, e.g., [1] pp. 212–216), we conclude that P_j is a factor of P. This is summarized as follows:

Theorem 2 *The degree of the problem of evaluating a predicate expressed by the sign of a polynomial P is the maximum arithmetic degree of the factors of P that are not always positive or always negative.*

As an application of Theorem 2, we can prove a lower bound on the degree of the nearest neighbor search problem.

Theorem 3 *The nearest neighbor search problem for a point set has degree* 2 *in any fixed dimension* $d \geq 2$.

Observe that an optimal-degree algorithm for the nearest neighbor search problem in a planar point set can be easily obtained with the brute force approach, where one computes all the distances between the query point and all other points and reports the point at minimum distance. However, such algorithm is both computationally inefficient (it requires quadratic time) and does not support repetitive-mode queries. In Section 3, we present an optimal degree algorithm whose query time and space are optimal.

Let q be a query point, let p_1 and p_2 be two points, and let r_1 and r_2 be two lines on the plane. The point-to-points distance test determines whether q is closer to p_1 or to p_2. The point-to-point-line distance test determines whether q is closer to p_1 or to r_1. The point-to-lines distance test determines whether q is closer to r_1 or to r_2.

Lemma 2 *The* point-to-points distance test, point-to-point-line distance test, point-to-lines distance test *can be solved with degree 2, 4, and 6, respectively.*

Another fundamental proximity primitive is the incircle test, that is testing whether the circle determined by three distinct sites (points and segments) of the plane contains a given query site. An incircle test is expressed as a quadruple $(a_1, a_2, a_3; a_4)$, where each $a_i \in \{p, l\}$ $(i = 1, \ldots, 4)$ is either a point or a line on the plane and we test whether a_4 intersects the circle determined by a_1,a_2, and a_3. The incircle test can be generalized to the insphere test in any dimension $d \geq 2$. The insphere test $(p_1, \ldots, p_{d+1}; p_{d+2})$, where points $p_1, \ldots, p_{d+1}$ determine a d-dimensional sphere and p_{d+2} is the query point.

Lemma 3 [2] *The* incircle test $(l_1, l_2, l_3; l_4)$ *can be solved with degree* 40. *The* insphere test $(p_1, \ldots, p_{d+1}; p_{d+2})$ *in any fixed dimension* $d \geq 2$ *can be solved with degree* $d + 2$.

3 Point Sites in the Plane

In this section, under our standard assumption that all input parameters — such as coordinates of sites and query points — are represented by b-bit integers, we consider the following proximity queries on a set S of point sites in the plane:

nearest neighbor search: given query point q, find a site of S whose Euclidean distance from q is less than or equal to that of any other site;

k-nearest neighbors search: given query point q, find k sites of S whose Euclidean distances from q are less than or equal to that of any other site;

circular range search: given query points q and r, find the sites of S that are inside the circle with center q passing through r.

It is well known that such queries are efficiently solved by performing point location in the Voronoi diagram (possibly of higher order) $V(S)$ of the sites [22].

The ***chain method*** [19], the ***bridged chain method*** [10], the ***trapezoid method*** [21], the ***subdivision refinement method*** [18], and the ***persistent search tree method*** [24] are popular deterministic techniques for point location in a planar map that combine theoretical efficiency with good performance in practice (see, e.g., [22]). The ***randomized-incremental method*** [14] also exists, that is specialized for point location in Voronoi diagrams.

By a careful examination of the query algorithms used in the point location methods presented in the literature, it is possible to clearly separate the primitive operations that access the geometry of the map from those that access only the topology. We say that a point location method is ***native*** for a class of maps if it performs point locations queries in a map M of the class by accessing the geometry of M exclusively through the following three geometric test primitives that discriminate the query point with respect to the vertices and edges of M:

above-below(q, v) determine whether query point q is vertically above or below vertex v.

left-right(q, v) determine whether query point q is horizontally to the left or to the right of vertex v.

left-right(q, e) determine whether query point q is to the left or to the right of edge e; this operation assumes that edge e is not horizontal and its vertical span includes q.

Test primitive left-right(q, v) is typically used only in degenerate cases (e.g., in the presence of horizontal edges).

Some point location methods work on modified versions of the original subdivision by means of auxiliary geometric objects introduced in the preprocessing (e.g., triangulation or regularization edges). We say that a point location method is ***ordinary*** for a class of maps if it performs point locations queries in a map M of the class by accessing the geometry of M through the three geometric test primitives described above for the native methods and through left-right(q, e) tests such that e is a fictitious edge connecting two vertices of M.

More specifically, we have:

Lemma 4 *The trapezoid method and the persistent search tree method are native for general maps. The chain method and the bridged chain method are ordinary for general maps and native for monotone maps. The subdivision refinement method is ordinary for general maps. The randomized-incremental method is ordinary for Voronoi diagrams.*

Hence, all the known planar point location methods described in the literature are ordinary for Voronoi diagrams.

Let S be a set of n point sites in the plane, where each site is a primitive point with b-bit integer coordinates. The Voronoi diagram $V(S)$ of S is said to be ***explicit*** if the coordinates of the vertices of $V(S)$ are computed and stored with exact arithmetic, i.e., as rational numbers (pairs of integers).

Lemma 5 *The* left-right(q, e) *test primitive in an explicit Voronoi diagram of point sites in the plane has degree* 6.

An algorithm for proximity queries on a set S of point sites in the plane is said to be ***conventional*** if it computes the explicit Voronoi diagram $V(S)$ of S and then performs point location queries on $V(S)$ with an ordinary method. Note that the class of conventional proximity query algorithms includes all the efficient algorithms presented in the literature. A conventional proximity query algorithm needs to perform test primitive left-right(q, e). Thus, by Lemma 5 we have:

Theorem 4 *Conventional algorithms for the following proximity query problems on a set of point sites in the plane have degree at least* 6*: (i) nearest neighbor query; (ii) k-nearest neighbor query; and (iii) circular range query.*

We observe that a degree-6 algorithm implies that a k-bit arithmetic unit can handle with native precision queries for points in a grid of size at most $2^{k/6} \times 2^{k/6}$. For example, if $k = 32$, the points that can be treated with single-precision arithmetic belong to a grid of size at most 64×64.

The extremal-search method [11], also designed for proximity queries, reduces the nearest neighbor search problem for a set S of 2D point sites to an extremal search problem.

Theorem 5 *The extremal-search method for the nearest neighbor query problem on a set of point sites in the plane has degree at least* 4.

Let S be a set of n point sites in the plane, and recall our assumption that each site or query point is a primitive point with b-bit integer coordinates. We say that a number s is a *semi-integer* if it is a rational number of the type $s = m/2$ for some integer m. The *implicit Voronoi diagram* $V^*(S)$ of S is a representation of the Voronoi diagram $V(S)$ of S that consists of a topological component and of a geometric component. The topological component of $V^*(S)$ is the planar embedding of $V(S)$, represented by a suitable data structure (e.g., doubly-connected edge lists [22] or the data structure of [15]). The geometric component of $V^*(S)$ stores the following geometric information for each vertex and edge of the embedding:

- For each vertex v of $V(S)$, $V^*(S)$ stores semi-integers $x^*(v)$ and $y^*(v)$ that approximate the x- and y-coordinates $y(v)$ of v, We provide the definition of $y^*(v)$ below. The definition of $x^*(v)$ is analogous.

$$y^*(v) = \begin{cases} y(v) & 0 \leq y(v) \leq 2^b - 1,\ y(v) \text{ integer} \\ \lfloor y(v) \rfloor + \frac{1}{2} & 0 \leq y(v) \leq 2^b - 1,\ y(v) \text{ not integer} \\ 2^b - \frac{1}{2} & y(v) > 2^b - 1 \\ 0 & y(v) < 0 \end{cases}$$

  Note that semi-integers $x^*(v)$ and $y^*(v)$ can be stored with $(b+1)$-bits.

- For each non-horizontal edge e of $V(S)$, $V^*(S)$ stores the pair of sites $\ell(e)$ and $r(e)$ such that e is a portion of the perpendicular bisector of $\ell(e)$ and $r(e)$, and $\ell(e)$ is to the left of $r(e)$.

Equipped with the above information, the three test primitives for point location can be performed in the implicit Voronoi diagram $V^*(S)$ as follows:

above-below(q, v) compare the y-coordinate of q with $y^*(v)$;

left-right(q, v) compare the x-coordinate of q with $x^*(v)$;

left-right(q, e) compare the Euclidean distances of point q from sites $\ell(e)$ and $r(e)$.

Since the query point q is by assumption a primitive point whose coordinates are b-bit integers, we have that $y(q) \leq y(v)$ if and only if $y(q) \leq y^*(v)$, where testing the latter inequality has degree 1. Similar considerations apply to testing $x(q) \leq x(v)$. This proves the correctness of our implementation of above-below(q, v) and left-right(q, v).

The correctness of the above implementation of test left-right(q, e) follows directly from the definition of Voronoi edges. Thus, in an implicit Voronoi diagram, test left-right(q, e) can be implemented with a point-to-points distance test that has degree 2 (Lemma 2).

Hence, we obtain the following lemma:

Lemma 6 *In an implicit Voronoi diagram of point sites in the plane test primitives* above-below(q, v) *and* left-right(q, v) *can be performed in $O(1)$ time with degree 1, and test primitive* left-right(q, e) *in $O(1)$ time with degree 2.*

In order to execute a native point location algorithm in an implicit Voronoi diagram, we only need to redefine the implementation of the three test primitives. By having encapsulated the geometry in the test primitives, no further modifications are needed. Hence, by Lemma 6 and Theorem 3 we obtain:

Lemma 7 *For any native method on a class of maps that includes Voronoi diagrams, a point location query in an implicit Voronoi diagram has optimal degree 2 and has the same asymptotic time complexity as a point location query in the corresponding explicit Voronoi diagram.*

In order to compute the implicit Voronoi diagram $V^*(S)$, we begin by constructing the Delaunay triangulation of S, denoted $DT(S)$, by means of the $O(n \log n)$-time algorithm of [15], which has degree 4 because its most expensive operation in terms of the degree is the incircle test (see Lemma 3). The topological structure of $V(S)$ and the sites $\ell(e)$ and $r(e)$ for each edge e of $V(S)$ are immediately derived from $DT(S)$ by duality. Next, we compute the approximations $x^*(v)$ and $y^*(v)$ for each vertex v of $V(S)$ by means of integer division. Let a, b, and c be the three sites of S that define vertex v. Adopting the same notation as in the proof of Lemma 5, the y-coordinate $y(v)$ of v is given by the ratio $y(v) = \frac{Y_1}{2W_1}$, where Y_1 is a variable of arithmetic degree 3 and W_1 is a variable of arithmetic degree 2, and similarly for $x(v)$. Hence, the computation of $x^*(v)$ and $y^*(v)$ involves integer represented by at most $3(b+O(1))$ bits. We summarize the above analysis as follows.

Lemma 8 *The implicit Voronoi diagram of n point sites in the plane can be computed in $O(n \log n)$ time, $O(n)$ space, and with degree 4.*

Theorem 6 *Let S be a set of n point sites in the plane. There exists an $O(n)$-space data structure for S, based on the implicit Voronoi diagram $V^*(S)$, that can be computed in $O(n \log n)$ time with degree* 5, *and supports nearest neighbor queries in $O(\log n)$ time with optimal degree* 2.

We can define implicit higher order Voronoi diagrams for point sites in the plane, and extend the above results to k-nearest neighbors and circular range search queries. We have

Theorem 7 *Let S be a set of n point sites in the plane and k an integer with $1 \leq k \leq n-1$. There exists an $O(k(n-k))$-space data structure for S, based on the implicit order-k Voronoi diagram $V_k^*(S)$, that can be computed in $O(k(n-k)\sqrt{n} \log n)$ time with degree* 5 *and supports k-nearest neighbors queries in $O(\log n + k)$ time with optimal degree* 2.

Theorem 8 *Let S be a set of n point sites in the plane. There exists an $O(n^3)$-space data structure for S, based on implicit order-k Voronoi diagrams, that can be computed in $O(n^3)$ time with degree 5 and supports circular range search queries in $O(\log n \log\log n + k)$ time with optimal degree 2.*

4 Point Sites in 3D Space

In this section, we consider the following proximity query on a set S of point sites in three-dimensional (3D) space:

nearest neighbor search: given query point q, find a site of S whose Euclidean distance from q is less than or equal to that of any other site;

We recall our assumption that the sites and query points are primitive points represented by b-bit integers.

As for the two-dimensional case, such query is efficiently answered by performing point location in the 3D Voronoi diagram of S.

There are only two known efficient spatial point location methods for cell-complexes that are applicable to 3D Voronoi diagrams: the *separating surfaces method* [5, 26], which extends the chain-method [19], and the *persistent planar location method* [23], which extends the persistent search tree method [24]. Let N be the number of facets of a cell-complex C. The query time is $O(\log^2 N)$ for both methods. The space used is $O(N)$ for the separating surfaces method and $O(N \log^2 N)$ for the persistent planar location method. Both methods are restricted to convex cell-complexes. The separating surfaces method is further restricted to acyclic convex cell-complexes, where the dominance relation among cells in the z-direction is acyclic.

We can separate the primitive operations that access the geometry of the cell-complex from those that access only the topology. We say that a point location method is *native* for a class of 3D cell-complexes if it performs point locations queries in a cell complex C of the class by accessing the geometry of C exclusively through the following three geometric test primitives that discriminate the query point with respect to the vertices and edges of C:

above-below(q, v) compare the z-coordinate of the query point q with the z-coordinate of vertex v.

left-right(q, v) compare the x-coordinate of the query point q with the x-coordinate of vertex v.

front-rear(q, v) compare the y-coordinate of the query point q with the y-coordinate of vertex v.

left-right(q_{xy}, e_{xy}) compare the xy-projection q_{xy} of the query point q with the xy-projection of edge e_{xy}. This operation assumes that e_{xy} is not parallel to the x-axis and its y-span includes q_{xy}.

above-below(q, f) determine whether query point q is above or below a facet f; this operation assumes that facet f is not parallel to the z-axis and that the xy-projection of f contains the xy-projection of q.

Test primitives above-below(q, v) and left-right(q, v) are used only in degenerate cases (e.g. in the presence of facets parallel to the z-axis and in cases where e_{xy} is horizontal).

Lemma 9 *The separating surfaces method is native for acyclic convex cell-complexes. The persistent planar location method is native for convex cell-complexes.*

Hence, all the known spatial point location methods described in the literature are native for 3D Voronoi diagrams.

Let S be a set of n point sites in 3D, where each site is a primitive point with b-bit integer coordinates. The 3D Voronoi diagram $V(S)$ of S is said to be *explicit* if the coordinates of the vertices of $V(S)$ are computed and stored with exact arithmetic, i.e., as rational numbers (pairs of integers).

Lemma 10 *The* left-right(q_{xy}, e_{xy}) *test primitive in an explicit Voronoi diagram of point sites in 3D space has degree* 8.

An algorithm for nearest neighbor queries on a set S of point sites in 3D space is said to be *conventional* if it computes the explicit 3D Voronoi diagram $V(S)$ of S and then performs point location queries on $V(S)$ with a native method. Recall that the class of conventional nearest neighbor query algorithms includes the two efficient algorithms presented in the literature. A conventional proximity query algorithm needs to perform test primitive left-right(q_{xy}, e_{xy}). Thus, by Lemma 10, we have:

Theorem 9 *Conventional algorithms for the nearest neighbor query problem on a set of point sites in 3D space have degree at least* 8.

The definition of the implicit 3D Voronoi diagram $V^*(S)$ of a set of S of point sites in 3D space is a straightforward extension of the definition for two-dimensional Voronoi diagrams given in Section 3. Namely $V^*(S)$ stores the topological structure of the 3D Voronoi diagram $V(S)$ of S (e.g., the data structure of [9]) and the following geometric information for each vertex and facet:

- For each vertex v of $V(S)$, $V^*(S)$ stores the semi-integer $(b+1)$-bit approximations $x^*(v)$, $y^*(v)$ and $z^*(v)$ of the x-, y-, and z-coordinates of v.

- For each facet f of $V(S)$ that is not parallel to any of three Cartesian planes, $V^*(S)$ stores the pair of sites $\ell(f)$ and $r(f)$ such that f is a portion of the perpendicular bisector of $\ell(f)$ and $r(f)$, and $\ell(f)$ is below $r(f)$.

The tests above-below(q, v), left-right(q, v), front-rear(q, v) can be implemented comparing the x-, y- and z-coordinate of query point q with $x(v)^*$, $y(v)^*$, and $z(v)^*$ respectively. With the same reasoning as for the two-dimensional case (see Section 3), it is easy to see that such implementations are correct. Test primitive above-below(q, f) is implemented by comparing the Euclidean distances of point q from the two sites $\ell(e)$ and $r(e)$ of which f is the perpendicular bisector, with a point-to-points distance test. The implementation is correct by the definition of Voronoi facet. Thus, by Lemma 2, we have.

Lemma 11 *In an implicit Voronoi diagram of 3D point sites test primitives* above-below(q, v), left-right(q, v), front-rear(q, v) *can be performed in* $O(1)$ *time with degree* 1, *and test primitive* above-below(q, f) *can be performed in* $O(1)$ *time and with degree* 2.

Finally, test left-right(q_{xy}, e_{xy}) is implemented by determining the sign of the equation of the line that contains edge e_{xy} when computed at point q_{xy}.

Lemma 12 *Test primitive* left-right(q_{xy}, e_{xy}) *in an implicit Voronoi diagram of 3D point sites can be performed in* $O(1)$ *time and with degree* 3.

In order to execute a native point location algorithm in an implicit 3D Voronoi diagram, we only need to redefine the implementation of the five test primitives. By having encapsulated the geometry in the test primitives, no further modifications are needed. Hence, by Lemmas 11–12 we obtain:

Lemma 13 *For any native method on a class of cell-complexes that includes 3D Voronoi diagrams, a point location query in an implicit 3D Voronoi diagram has degree 3 and has the same asymptotic time complexity as a point location query in an explicit 3D Voronoi diagram.*

The Voronoi diagram of n point sites in 3D space is an acyclic convex cell complex with $N = O(n^2)$ facets. Hence, using the separating surfaces method on the implicit 3D Voronoi diagram yields the following result:

Theorem 10 *Let S be a set of n point sites in 3D space. There exists an $O(n^2)$-space data structure for S that can be computed in $O(n^2)$ time with degree* 7 *and supports nearest neighbor queries in $O(\log^2 n)$ time with degree* 3.

Although the algorithm for nearest neighbor queries proposed in this section has nonoptimal degree 3, it is a practical approach for the important application scenario where the primitive points are pixels on a computer screen. On a typical screen with about $2^{10} \times 2^{10}$ pixels, our nearest neighbor query can be executed with the standard integer arithmetic of a 32-bit processor.

5 Point and Segment Sites in the Plane

In this section, we consider the following proximity query on a set S of point and segment sites in the plane:

nearest neighbor search: given query point q, find a site of S whose Euclidean distance from q is less than or equal to that of any other site.

As for the other queries studied in the previous sections, such query is efficiently solved by performing point location in the Voronoi diagram of the set of point and segment sites [22]. The Voronoi diagram $V(S)$ of a set S of point and segment sites is a map whose edges are either straight-line segments or arcs of parabolas. Hence, in general $V(S)$ is neither convex nor monotone. In order to perform point location in $V(S)$, we refine $V(S)$ into a map with monotone edges as follows. If edge e of $V(S)$ is an arc of parabola whose point p of maximum (or minimum) y-coordinate is not a vertex, we split e into two edges by inserting a fictitious vertex at point p. We call the resulting map the *extended Voronoi diagram* $V'(S)$ of S. The persistent search tree method and the trapezoid method can be used as native methods on the extended Voronoi diagram, where the test primitives are the same as those defined in Section 3 for point sites. If we want to use the chain method or the bridged chain method, we need to do a further refinement that transforms the map into a monotone map by adding vertical fictitious edges emanating from the fictitious vertices previously inserted along the parabolic edges.

Lemma 14 *The trapezoid method and the persistent search tree method are native, and the chain method and the bridged chain method are ordinary for extended Voronoi diagrams of point and segment sites.*

Let S be a set of n points and segment sites. The extended Voronoi diagram $V'(S)$ of S is said to be *explicit* if the coordinates of the vertices of $V'(S)$ are computed and stored with exact arithmetic, i.e., as algebraic numbers [3, 28]. In the following lemma, we analyze the degree of test primitive left-right(q, e) for a straight-line edge e of an explicit extended Voronoi diagram.

Lemma 15 *The* left-right(q, e) *test primitive for a straight-line edge e in an explicit extended Voronoi diagram of point and segment sites in the plane has degree* 64.

An algorithm for proximity queries on a set S of point and segment sites in the plane is said to be *conventional* if it computes the explicit extended Voronoi diagram $V'(S)$ of S and then performs point location queries on $V'(S)$ with a native method. Note that the class of conventional proximity query algorithms includes all the efficient algorithms presented in the literature. A conventional proximity query algorithm needs to perform test primitive left-right(q, e). Thus, by Lemma 15 we conclude:

Theorem 11 *Conventional algorithms for the nearest neighbor query problem on a set of point and segment sites in the plane have degree at least* 64.

Our analysis shows that performing point location in an explicit Voronoi diagram of points and segments is not practically feasible due to the high degree.

The definition of the implicit Voronoi diagram $V^*(S)$ of a set of S of point and segment sites is a straightforward extension of the definition for Voronoi diagrams of point sites given in Section 3. Namely $V^*(S)$ stores the topological structure of the extended Voronoi diagram $V'(S)$ of S (e.g., the data structure of [9]) and the following geometric information for each vertex and edge:

- For each vertex v of $V'(S)$, $V^*(S)$ stores the semi-integer $(b+1)$-bit approximations $x^*(v)$ and $y^*(v)$ of the x- and y-coordinates of v.
- For each non-horizontal edge e of $V'(S)$, $V^*(S)$ stores the pair of sites $\ell(e)$ and $r(e)$ such that e is a portion of the bisector of $\ell(e)$ and $r(e)$, and $\ell(e)$ is to the left of $r(e)$.

In the implicit Voronoi diagram $V^*(S)$ of S, test left-right(q, e) is implemented by comparing the distances of query point q from sites $\ell(e)$ and $r(e)$ with one of the following tests, depending on the type (point or line) of sites $\ell(e)$ and $r(e)$: point-to-lines distance test, point-to-point-line distance test, or point-to-points distance test. Thus, by Lemma 2, we have.

Lemma 16 *For any native method on a class of maps that includes extended Voronoi diagrams of point and segment sites in the plane, a point location query in an implicit Voronoi diagram has degree 6 and has the same asymptotic time complexity as a point location query in an explicit Voronoi diagram.*

Theorem 12 *Let S be a set of n point and segment sites in the plane. There exists an $O(n)$-space data structure for S that can be computed in $O(n \log n)$ expected time with degree* 40 *and supports nearest neighbor queries in $O(\log n)$ time with degree* 6.

6 Simplified Implicit Voronoi Diagrams

In this section, we describe a modification of implicit Voronoi diagrams of point sites that allows us to reduce the degree of the preprocessing task from 5 to 4 when the sites are in the plane (see Theorems 6—8), and from 7 to 5 when the sites are in three-dimensional space (see Theorem 10). This modification also has a positive impact on the space requirement of the data structure and on the running time of point location queries. Let $V(S)$ be the Voronoi diagram of a set S of point sites in the plane. We recall our standard assumption that all input parameters — such as coordinates of sites and query points — are represented as b-bit integers. An *island* of $V(S)$ is a connected component of the map obtained from $V(S)$ by removing all the vertices with integer y-coordinate and all the edges containing a point with integer y-coordinate. Note that for any two vertices v_1 and v_2 of an island, $y^*(v_1) = y^*(v_2) = m + \frac{1}{2}$ for some integer m, where $y^*(v)$ is the semi-integer approximation defined in Section 3. The *simplified implicit Voronoi diagram* $V^\circ(S)$ of S is a representation of the Voronoi diagram $V(S)$ of S that consists of a topological component and of a geometric component. The topological component of $V^\circ(S)$ is the planar embedding obtained from $V(S)$ by contracting each island of $V(S)$ into an *alias vertex*. The geometric component of $V^\circ(S)$ stores the following geometric information for each vertex and edge of the embedding:

- For each vertex v that is also a vertex of $V(S)$, $V^\circ(S)$ stores the $(b+1)$-bit semi-integers approximations $x^*(v)$ and $y^*(v)$.
- For each alias vertex a, which is associated with an island of $V(S)$, $V^\circ(S)$ stores semi-integer $y^*(a)$ such that $y^*(a) = y^*(v)$ for each vertex v of the island.
- For each non-horizontal edge e that is also an edge of $V(S)$, $V^\circ(S)$ stores the pair of sites $\ell(e)$ and $r(e)$ such that e is a portion of the perpendicular bisector of $\ell(e)$ and $r(e)$, and $\ell(e)$ is to the left of $r(e)$.

The space requirement of the simplified implicit Voronoi diagram is less than or equal to that of the implicit Voronoi diagram, since each island is represented by a single alias vertex storing only its semi-integer y-approximation. We can show examples where the simplified implicit Voronoi diagram of n point sites has $O(n)$ fewer vertices and edges than the corresponding implicit Voronoi diagram.

Lemma 17 *For any native method on a class of maps that includes monotone maps, a point location query in a simplified implicit Voronoi diagram has optimal degree 2 and executes a number of operations less than or equal to a point location query in the corresponding explicit Voronoi diagram.*

Lemma 18 *The simplified implicit Voronoi diagram of n point sites in the plane can be computed in $O(n \log n)$ time, $O(n)$ space, and with degree 4.*

The main advantage of the simplified implicit Voronoi diagram with respect to the degree cost measure is that the additional test primitive needed in the preprocessing that consists of comparing the y-coordinates of two Voronoi vertices (see the proof of Theorem 6) is now reduced to the comparison of two $(b+1)$-bit semi-integers, and thus has degree 1. Hence, the preprocessing for point location using a native method for monotone maps has degree 1.

Theorem 13 *Let S be a set of n point sites in the plane. There exists an $O(n)$-space data structure for S, based on the simplified implicit Voronoi diagram $V^{\circ}(S)$, that can be computed in $O(n \log n)$ time with degree 4 and supports nearest neighbor queries in $O(\log n)$ time with optimal degree 2.*

Using a similar approach, we can define simplified implicit order-k Voronoi diagrams for point sites in the plane and simplified implicit Voronoi diagrams for point sites in three-dimensional space. This reduces the degree of the preprocessing from 5 to 4 in Theorems 7 and 8, and from 7 to 5 in Theorem 10.

References

[1] M. Bocher. *Introduction to higher algebra.* Macmillan, 1907.

[2] C. Burnikel. *Exact Computation of Voronoi Diagrams and Line Segment Intersections.* Ph.D thesis, Universität des Saarlandes, Mar. 1996.

[3] C. Burnikel, J. Könnemann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Exact geometric computation in LEDA. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages C18–C19, 1995.

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

[5] B. Chazelle. How to search in history. *Inform. Control*, 64:77–99, 1985.

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

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

[8] D. P. Dobkin and D. G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoret. Comput. Sci.*, 27:241–253, 1983.

[9] D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32, 1989.

[10] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15:317–340, 1986.

[11] H. Edelsbrunner and H. A. Maurer. Finding extreme points in three dimensions and solving the post-office problem in the plane. *Inform. Process. Lett.*, 21:39–47, 1985.

[12] S. Fortune. Stable maintenance of point set triangulations in two dimensions. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 494–505, 1989.

[13] S. Fortune and C. J. Van Wyk. Efficient exact arithmetic for computational geometry. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 163–172, 1993.

[14] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[15] 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.

[16] C. M. Hoffmann. The problems of accuracy and robustness in geometric computation. *IEEE Computer*, 22(3):31–41, Mar. 1989.

[17] M. Karasick, D. Lieber, and L. R. Nackman. Efficient Delaunay triangulations using rational arithmetic. *ACM Trans. Graph.*, 10:71–91, 1991.

[18] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12:28–35, 1983.

[19] D. T. Lee and F. P. Preparata. Location of a point in a planar subdivision and its applications. *SIAM J. Comput.*, 6:594–606, 1977.

[20] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries: an illustration of degree-driven algorithm design. *SIAM J. Comput.* (to appear). Also available as Technical Report CS-96-16, Center for Geometric Computing, Comput. Sci. Dept., Brown Univ., Providence, RI, 1996. URL: http://www.cs.brown.edu/cgc/cgc-papers/cgc-papers.html.

[21] F. P. Preparata. A new approach to planar point location. *SIAM J. Comput.*, 10:473–482, 1981.

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

[23] F. P. Preparata and R. Tamassia. Efficient point location in a convex spatial cell-complex. *SIAM J. Comput.*, 21:267–280, 1992.

[24] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29:669–679, 1986.

[25] K. Sugihara and M. Iri. Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic. *Proc. IEEE*, 80(9):1471–1484, Sept. 1992.

[26] R. Tamassia and J. S. Vitter. Optimal cooperative search in fractional cascaded data structures. *Algorithmica*, 15(2), 1996.

[27] C. K. Yap. Symbolic treatment of geometric degeneracies. *J. Symbolic Comput.*, 10:349–370, 1990.

[28] C. K. Yap. Toward exact geometric computation. *Computational Geometry: Theory and Applications* (to appear).