SIAM J. COMPUT.
Vol. 28, No. 3, pp. 864–889

ROBUST PROXIMITY QUERIES: AN ILLUSTRATION OF DEGREE-DRIVEN ALGORITHM DESIGN*

GIUSEPPE LIOTTA†, FRANCO P. PREPARATA‡, AND 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 where algorithm design is driven also by the degree, we consider the important classical problem of proximity queries in two and three 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 offers 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.

Key words. geometric computing, robustness, arithmetic precision, proximity queries

AMS subject classifications. 68U05, 65Y25

PII. S0097539796305365

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., [4, 12, 11, 19, 27, 28, 31, 36, 34, 39, 48, 54, 58, 21, 30, 32]). For an early survey of the different approaches to robust computational geometry the reader is referred to [38].

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

*Received by the editors June 19, 1996; accepted for publication (in revised form) January 17, 1997; published electronically September 22, 1998. This research was supported in part by U.S. Army Research Office grant DAAH04–96–1–0013, National Science Foundation grant CCR–9423847, the N.A.T.O.-C.N.R. Advanced Fellowships Programme, and EC ESPRIT Long Term Research Project ALCOM-IT contract 20244.

http://www.siam.org/journals/sicomp/28-3/30536.html

†Dipartimento di Informatica e Sistemistica, Universitá di Roma "La Sapienza", Via Salaria 113, Roma I-00198, Italy. The work of this author was performed in part while 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 (franco@cs.brown.edu, rt@cs.brown.edu).

Yap [59] and by the Saarbrücken school [10], and so does our approach. Within this paradigm, we introduce the notion of *degree* of an algorithm, which describes, up to a small additive constant, the arithmetic precision (i.e., number of bits) needed by the exact-computation paradigm. Namely, if the coordinates of the input points of a degree-d geometric algorithm are b-bit integers, then, as we shall substantiate below, the algorithm may be required in 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., [41, 49]). 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 reexamining the entire rich body of computational geometry as we know it today.

In this paper, we consider as a test case a problem area, geometric proximity, which plays a major role in several 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., [9, 11, 20, 29, 31, 27, 55, 32]). In particular we shall illustrate the role played by the degree criterion if one wishes to comply with the standard exact-computation paradigm.

To illustrate the approach, we recall that Voronoi diagrams are the search structures which permit answering a proximity query without evaluating all query/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 supports 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 [26]). However, we shall show that the latter method fails to achieve optimal degree because the search

Table 1

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

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	$O(\log^2 n)$

is based on predicates requiring $4b$ bits of precision; moreover, the high overhead of the search technique (which uses the hierarchical polytope representation [22]) casts some doubts on the practicality of the method.

The main results of this work are summarized in Table 1. Considering, for the time being, the degree as a measure of complexity, we show that previous methods 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.

The rest of this paper is organized as follows. In section 2, the concept of degree of a geometric algorithm is defined and a simple formalism to compute such degree is introduced. Such formalism is used in section 3 to analyze the performance of basic proximity primitives. In section 4, we consider the following fundamental proximity queries for a set of point sites in the plane: nearest neighbor search, k-nearest neighbors search, and circular range search. We show that the existing methods for efficiently answering such queries have degree either 6 (point location in explicit Voronoi diagram) or 4 (extremal-search method), and we present our new technique, based on implicit Voronoi diagrams, which achieves optimal degree 2. In sections 5–6, we extend our approach to nearest neighbor search queries in a set of 3D point sites and in a set of point and segment sites in the plane, respectively. Practical improvements are presented in section 7. Finally, further research directions are discussed in section 8.

2. Degree of geometric algorithms and 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. The 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 as $\{+, -, \times, \div, \sqrt[2]{}, \dots\}$. As we shall show below, the expressions under consideration are equivalent to multivariate polynomials.

Here we make 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 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.*

Remark 1. Recently, Burnikel [9] 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 [58, 59]. Given a set of numbers, any number x of the set has depth 0. A number has depth at most d if it can be obtained by executing a rational operation on numbers with depth $d-1$ or it is the result of a root extraction from a degree-k polynomial whose coefficients have depth at most $d-k$. An algorithm has *depth d* if it performs only rational operations such that all the intermediate computed numbers have depth of derivation at most d with respect to the set of input numbers. Clearly, d is the least possible integer such that all the intermediate computed values have depth of derivation at most d. A problem has *depth d* if it can be solved by an algorithm with rational bounded depth d. Despite the relatedness of the notions of depth and degree, the latter seems more appropriate to our analysis, where we aim at minimizing the number of bits needed for computing an exact value, independently of its (possibly very high) depth.

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 radical). 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

$$E_1 \geq E_2 \Longleftrightarrow (-1)^{\sigma(D_1)+\sigma(D_2)}(N_1 D_2 - N_2 D_1) \geq 0,$$

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 nonradicals. 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

$$E \geq 0 \Longleftrightarrow E''R \geq -E'.$$

The resulting expression $({E''}^2 R^2 - {E'}^2)$ does not contain R. Therefore, by this device, referred to as *segregate and square*, 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. Besides these conventional algebraic rules, we have a set of *rewriting rules* of the form $A \rightarrow B$, meaning that the sign of A is *unambiguously determined* by the sign of B and by the signs of terms in A, which are inductively assumed to be known. This induction is either on the degree of the terms or, in case of addition of (same degree) terms, on the number of the latter.

We have seven rules, whose correctness can be proved with elementary algebra. Rule 1 performs genericization, i.e., a specific term ρ_j, which is known to be a polynomial of degree s over primitive variables, can be rewritten as α^s. Rules 2–4 involve generic terms, which reflect the fact that the only relevant feature of a polynomial is its degree. Finally, rules 5–7 concern specific terms. The role of specific terms is that we wish to keep track of their structure (that is, their definition) in order to exploit it when computing least common multiples or multiplying radicals together. Again, the R.H.S. of a rule gives the highest degree of the polynomials whose signs unambiguously determine the sign of the L.H.S. Recall that the stated hypothesis of nonnegative dimensionality implies that the degree of a numerator is never smaller than that of its denominator. The rules are

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

A discussion on how to compute the sign of an algebraic expression of the type considered by rule (7) can also be found in [57].

The preceding discussion establishes the following theorem.

THEOREM 1. *Rules* (1)–(7) *are adequate to evaluate the degree of multivariate polynomials whose sign, collectively, unambiguously determines 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., [8, 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 a polynomial P is the maximum arithmetic degree of the factors of P that change sign over their domain.*

3. Basic proximity queries. In this section we use the formalism introduced above to analyze the degree of some geometric tests that answer basic proximity queries. We end the section by establishing a lower bound on the degree of the nearest neighbor search problem. In the proofs, we assume that a line r is represented by the coefficients of its equation. However, the results still hold if line r is represented by two of its points.

We start with the point-to-lines distance test; i.e., given two lines r_1 and r_2 on the plane and a query point q, determine whether q is closer to r_1 than to r_2.

LEMMA 2. *The* point-to-lines distance test *can be solved with degree* 6.

Proof. Let the equation of r_i be $a_ix + b_iy + c_i = 0$ $(i = 1, 2)$ and let $q \equiv (x_q, y_q)$. Then the test is to study the sign of $\frac{|a_1x_q+b_1y_q+c_1|}{\sqrt{a_1^2+b_1^2}} - \frac{|a_2x_q+b_2y_q+c_2|}{\sqrt{a_2^2+b_2^2}}$. By using the proposed notation, and with obvious meaning for ρ_1 and ρ_2, this test becomes (each arrow being superscripted with the rules used)

$$\frac{\alpha^2}{\rho_1} - \frac{\alpha^2}{\rho_2} \longrightarrow^{(6)} \alpha^2\rho_2 - \alpha^2\rho_1 \longrightarrow^{(7)} \alpha^4\rho_2^2 - \alpha^4\rho_1^2 \longrightarrow^{(1)}$$
$$\alpha^4\alpha^2 - \alpha^4\alpha^2 \longrightarrow^{(4,3)} \alpha^6. \qquad \square$$

The following lemmas describe the degree of other proximity primitives that will be useful in the rest of the paper. We omit the proofs of such lemmas, since they are either straightforward or have been already proved in [9]. However, it is worth mentioning that the proofs in [9] can be substantially simplified by using the proposed notation.

Let p be a point and r a line in the plane. The point-to-point-line distance test determines whether a query point q is closer to p or to r.

LEMMA 3. *The* point-to-point-line distance test *can be solved with degree* 4.

Let p_1 and p_2 be two distinct points of the plane and let q be a query point. The point-to-points distance test determines whether q is closer to p_1 or to p_2.

LEMMA 4. *The* point-to-points distance test *can be solved with degree* 2.

The above lemma can be easily extended to any space of dimension d.

Another fundamental proximity primitive is the incircle test, that is, testing whether the circle determined by three distinct sites (points and/or segments) of the plane contains a given query site. The incircle test is a basic operation for many algorithms that construct the Voronoi diagram of the sites (see, e.g., [37, 41, 33, 3]). The degree of the incircle test has been extensively studied by Burnikel [9] and by Burnikel, Mehlhorn, and Schirra [11]. Following the notation of Burnikel [9], an incircle test is conveniently 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 (a segment is seen by Burnikel as given by the pair of its endpoints and by the underlying line) and we test whether a_4 intersects the circle determined by a_1,a_2, and a_3.

The following lemma is proved observing that the incircle test $(p_1, p_2, p_3; p_4)$ can be answered by determining the sign of a 4×4 determinant that is an arithmetic degree-4 multivariate polynomial.

LEMMA 5 (see [9]). *The* incircle test $(p_1, p_2, p_3; p_4)$ *can be solved with degree* 4.

Lemma 5 can be easily extended to any dimension $d > 2$. We describe such a test as $(p_1, \ldots, p_{d+1}; p_{d+2})$, where points $p_1, \ldots, p_{d+1}$ determine a d-dimensional sphere and p_{d+2} is the query point.

LEMMA 6. *The* insphere test $(p_1, \ldots, p_{d+1}; p_{d+2})$ *in any fixed dimension* $d \geq 2$ *can be solved with degree* $d + 2$.

For the construction of the Voronoi diagram of a set of points and segments in the plane Burnikel shows that the most demanding test in terms of degree is the incircle test $(l_1, l_2, l_3; l_4)$ [9].

LEMMA 7 (see [9]). *The* incircle test $(l_1, l_2, l_3; l_4)$ *can be solved with degree* 40.

While the above lemmas provide an upper bound on the degree of a proximity problem, the next theorem gives a lower bound.

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

Proof. We show the proof for the case $d = 2$. The proof for any other values of d is analogous. Let $p_1 \equiv (x_1, y_1)$, $p_2 \equiv (x_2, y_2)$, and $q \equiv (x_q, y_q)$ be three points in the plane. In order to determine which of p_1 and p_2 is the point nearest to q, a point-to-points distance test must be performed.

This is equivalent to the evaluation of the sign of the difference $d(p_1, q) - d(p_2, q)$, which, in turn, is equivalent to the evaluation of the sign of the polynomial $d^2(p_1, q) - d^2(p_2, q)$. This shows that this computation has degree at most 2. On the basis of Theorem 2, for the degree to be less than 2, polynomial $d^2(p_1, q) - d^2(p_2, q)$ should be factorable as the product of two degree-1 polynomials. We show below that this is not possible.

Suppose, for a contradiction, that there exist constants $a', a'', b', b'', c', c'', d', d'', e, e'', f', f''$ such that

$$d^2(p_1, q) - d^2(p_2, q) = x_1^2 + y_1^2 - x_2^2 - y_2^2 - 2x_1x_q + 2x_2x_q - 2y_1y_q + 2y_2y_q$$
$$= (a'x_1 + b'y_1 + c'x_2 + d'y_2 + e'x_q + f'y_q) \cdot (a''x_1 + b''y_1 + c''x_2 + d''y_2 + e''x_q + f''y_q).$$

The above equality implies $e'e'' = 0$, since there cannot be a term $e'e''x_q^2$. However, e' and e'' are not simultaneously 0, because there are nonzero terms having x_q as a factor. Assume w.l.o.g. that $e'' \neq 0$. Observe that $d'e'' = 0$ because there is no term $d'e''y_2y_q$; this implies $d' = 0$. However, we must also have $d'd'' = -1$ because of the term $-y_2^2$, a contradiction. □

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 4 we present an optimal degree algorithm, complying with the exact-computation paradigm, whose query time and space are optimal.

4. Proximity queries for 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 [51]. For nearest neighbor search, the alternative extremal-search method [26] also exists.

We begin by examining in section 4.1 the geometric test primitives used by the theoretically optimal and practically efficient point location methods. We identify three fundamental geometric test primitives for accessing the geometry of a planar map, and we introduce the concepts of "native" and "ordinary" point location methods. In section 4.2, we show that the "conventional" approach of accessing the explicitly computed Voronoi diagram $V(S)$ of the sites causes point location queries, and hence proximity queries, to have degree at least 6. We also analyze the extremal-search method and show that it has degree 4. In sections 4.3–4.4, we describe our new implicit representation of Voronoi diagrams for point sites in the plane, which allows us to perform proximity queries with optimal degree 2.

4.1. Test primitives and methods for planar point location. The *chain method* [44], the *bridged-chain method* [25], the *trapezoid method* [50], the *subdivision refinement method* [42], and the *persistent search tree method* [53] are popular deterministic techniques for point location in a planar map that combine theoretical efficiency with good performance in practice (see, e.g., [24, 51]). Namely, denoting with n the size of the map, all the above point location methods require $O(n \log n)$ preprocessing time. The query time is $O(\log^2 n)$ for the chain method and $O(\log n)$ for the other methods. The space used is $O(n \log n)$ for the trapezoid method and $O(n)$ for the other methods. For monotone maps, the preprocessing time is $O(n)$ for the chain method and the bridged-chain method, and $O(n \log n)$ for the other methods. The *randomized-incremental method* [35] also exists. Such a method is specialized for point location in Voronoi diagrams, uses expected space $O(n)$, and has expected query time $O(\log^2 n)$.

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 location 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 location 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.

Now, we analyze the chain method [44] for point location in a monotone map M. A binary tree represents a balanced recursive decomposition of map M by means of vertically monotone polygonal chains covering the edges of M, called *separators*. A point location query consists of traversing a root-to-leaf path in this tree, where at each node we determine whether the query point q is to the left or to right of the separator associated with the node. The discrimination of point q with respect to a separator σ is performed in two steps:

1. we find the edge e of σ whose vertical span includes point q by means of binary search on the y coordinates of the vertices of σ, which consists of performing a sequence of a logarithmic number of above-below(q, v) tests;
2. we discriminate q with respect to σ by performing test left-right(q, e).

In the special case that separator σ has horizontal edges, the discrimination of point q with respect to σ uses also test primitive left-right(q, v). Hence, the chain method is native for monotone maps. For a map M that is not monotone, fictitious "regularization" edges are added to M and the point location in M is reduced to point location in the resulting refinement M' of M. Hence, the chain method is ordinary for general maps.

In the bridged-chain method [25], the technique of *fractional cascading* [17, 18] is applied to the sets of y-coordinates of the separators. Fractional cascading establishes "bridges" between the separator of a node and the separators of its children such that there are $O(1)$ vertices between any two consecutive bridges. Hence, except for the separator of the root, step 1 can be executed with $O(1)$ above-below(q, v) tests for the vertices between two consecutive bridges. The bridged-chain method is ordinary for general maps and native for monotone maps.

A similar analysis shows that all efficient point-location methods described in the literature are ordinary for general maps. More specifically, we have the following lemma.

LEMMA 8. *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*

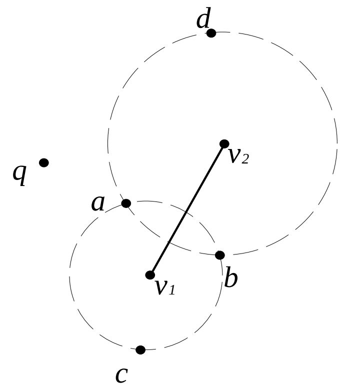

FIG. 1. *Illustration for Lemma* 9.

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.

4.2. Explicit 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 9. *The* left-right(q, e) *test primitive in an explicit Voronoi diagram of point sites in the plane has degree* 6.

Proof. Let $e \equiv (v_1, v_2)$ be a Voronoi edge such that $v_1 \equiv (x_1, y_1)$ is equidistant from three sites $a \equiv (x_a, y_a)$, $b \equiv (x_b, y_b)$, $c \equiv (x_c, y_c)$ and $v_2 \equiv (x_2, y_2)$ is equidistant from three sites $b \equiv (x_b, y_b)$, $c \equiv (x_c, y_c)$, and $d \equiv (x_d, y_d)$. See Figure 1. In an explicit Voronoi diagram, test primitive left-right(q, e) that determines whether query point $q \equiv (x_q, y_q)$ is to the left or to the right of edge $e \equiv (v_1, v_2)$ is equivalent to evaluating the sign of the following determinant:

$$\Delta = \begin{vmatrix} x_q & y_q & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = \begin{vmatrix} x_q & y_q & 1 \\ \frac{X_1}{2W_1} & \frac{Y_1}{2W_1} & 1 \\ \frac{X_2}{2W_2} & \frac{Y_2}{2W_2} & 1 \end{vmatrix} = \frac{1}{4W_1W_2} \begin{vmatrix} x_q & y_q & 1 \\ X_1 & Y_1 & 2W_1 \\ X_2 & Y_2 & 2W_2 \end{vmatrix} = \frac{\Delta'}{4W_1W_2},$$

where

$$X_1 = \begin{vmatrix} x_a^2 + y_a^2 & y_a & 1 \\ x_b^2 + y_b^2 & y_b & 1 \\ x_c^2 + y_c^2 & y_c & 1 \end{vmatrix}, \quad Y_1 = \begin{vmatrix} x_a & x_a^2 + y_a^2 & 1 \\ x_b & x_b^2 + y_b^2 & 1 \\ x_c & x_c^2 + y_c^2 & 1 \end{vmatrix}, \quad W_1 = \begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix}$$

and X_2, Y_2, and W_2 have similar expressions obtained replacing in the above determinants x_c with x_d and y_c with y_d. Evaluating the sign of Δ is equivalent to evaluating the signs of W_1, W_2 and of Δ'.

By using the notation introduced in section 2, we can rewrite X_i and Y_i as α^3, and W_i as α^2 $(i = 1, 2)$. Hence, Δ' is a degree-6 multivariate polynomial since it can be rewritten as

$$\alpha(\alpha^3\alpha^2 - \alpha^3\alpha^2) - \alpha(\alpha^3\alpha^2 - \alpha^3\alpha^2) + \alpha^3\alpha^3 - \alpha^3\alpha^3 \quad \longrightarrow^{(2,3,4)} \quad \alpha^6 + \alpha^6 \quad \longrightarrow^{(3)} \quad \alpha^6.$$

Although the explicit representation approach leads to Lemma 9, it should be noted that determinant Δ is a reducible polynomial,[1] one factor being the (always positive) incircle test polynomial of degree 4 for the four sites. □

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 9 we have the following theorem.

THEOREM 4. *Conventional algorithms for the following proximity query problems on a set of point sites in the plane have degree at least* 6*:*
- *nearest neighbor query,*
- *k-nearest neighbor query,*
- *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* [26], also designed for proximity queries, reduces the nearest neighbor search problem for a set S of 2D point sites to the following extremal-search problem. Let $\mathcal{P}$ be the paraboloid with equation $z = x^2 + y^2$, and let S' be the set of 3D points obtained by lifting S to $\mathcal{P}$. Given a query point q in the plane, let $\vec{r}$ be the unit vector orthogonal to the plane tangent to $\mathcal{P}$ at the lifted query point $q' \equiv (x_q, y_q, x_q^2 + y_q^2)$. The extremal-search problem for S' and query vector $\vec{r}$ consists of determining the first site s' of S' hit by a plane orthogonal to $\vec{r}$ translating from infinity toward S'. Projecting s' down onto the xy-plane gives the nearest neighbor s of q in S.

The extremal-search method makes use of 3D geometric primitives that guide the search through a data structure embodying the Dobkin–Kirkpatrick hierarchical representation [22] of the convex hull of S'. Such 3D geometric primitives in turn can be reduced to the following 2D geometric primitives:
- point-to-points distance test for q and a site of S, which has degree 2;
- the identification of suitably defined "extremal edges" of the Delaunay triangulation of a subset of S with respect to q.

The second primitive evaluates the sign of determinants of the type

$$\Delta = \begin{vmatrix} x_a & y_a & x_a^2 + y_a^2 \\ x_b & y_b & x_b^2 + y_b^2 \\ x_q & y_q & x_q^2 + y_q^2 \end{vmatrix},$$

where $a \equiv (x_a, y_a)$ and $b \equiv (x_b, y_b)$ are sites of S. By using the methodology introduced in section 2, we can show that Δ is a degree-4 multivariate polynomial. Thus, we have Theorem 5.

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.

4.3. Implicit Voronoi diagrams. 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

[1] K. Mehlhorn suggested that Δ was likely to be reducible.

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 [51] or the data structure of [37]). 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.

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

  Note that semi-integers $x^*(v)$ and $y^*(v)$ can be stored with $(b+1)$-bits.
- For each nonhorizontal 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 $y(q) \le y(v)$ if and only if $y(q) \le y^*(v)$, where testing the latter inequality has degree 1. Similar considerations apply to testing $x(q) \le x(v)$. 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 4).

Hence, we obtain the following lemmas.

LEMMA 10. *Test primitives* above-below(q, v) *and* left-right(q, v) *in an implicit Voronoi diagram of point sites in the plane can be performed in* $O(1)$ *time and with degree* 1.

LEMMA 11. *Test primitive* left-right(q, e) *in an implicit Voronoi diagram of point sites in the plane can be performed in* $O(1)$ *time and 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 Lemmas 10–11 we obtain Lemma 12.

LEMMA 12. *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 [37], which has degree 4 because its most expensive operation in terms of the degree is the incircle test (see Lemma 5). 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. For effective procedures that perform the integer division, see, e.g., *LEDA* [46]. 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 9, the y-coordinate $y(v)$ of v is given by the ratio $y(v) = \frac{Y_1}{2W_1}$, where Y_1 is a polynomial of degree 3 and W_1 is a polynomial of degree 2, and similarly for $x(v)$. Hence, the computation of $x^*(v)$ and $y^*(v)$ involves an integer represented by at most $3(b + O(1))$ bits. We summarize the above analysis as follows.

LEMMA 13. *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.

Proof. We perform point location in the implicit Voronoi $V^*(S)$ diagram of S using a native method for monotone maps with optimal space and query time such as the bridged-chain method or the persistent search tree method. The space requirement and the query degree and time follow from the performance of these methods and from Lemma 12.

Regarding the preprocessing time, by Lemma 13, the construction of the implicit Voronoi $V^*(S)$ takes $O(n \log n)$ time with degree 4. In order to construct the point location data structure, we also need an additional test primitive that consists of comparing the y-coordinates of two Voronoi vertices. For example, this primitive is used to establish bridges in the bridged-chain method (see section 4.1) and to sort the vertices by y-coordinate in the persistent location method. By using the same notation as in Lemma 9, comparing the y-coordinates of the Voronoi vertices is equivalent to evaluating the sign of multivariate polynomials of the form $\frac{Y_i}{2W_i} - \frac{Y_j}{2W_j}$, where $\frac{Y_i}{2W_i}$ and $\frac{Y_j}{2W_j}$ represent the y-coordinates of two different Voronoi vertices. Such multivariate polynomials have degree 5, since they can be rewritten as

$$\frac{\rho_i}{\rho_j} - \frac{\rho_h}{\rho_k} \quad \longrightarrow^{(6)} \quad \rho_i\rho_k - \rho_h\rho_j \quad \longrightarrow^{(1)} \alpha^3\alpha^2 - \alpha^3\alpha^2 \quad \longrightarrow^{(2,3,4)} \quad \alpha^5. \qquad \square$$

Remark 2. It must be pointed that for the problem under consideration similar results could be obtained by carrying out tests with limited accuracy, and therefore risking to mistakenly select a Voronoi site adjacent to the correct one in critical situations (when the query point is very close to the separating edge): such indeterminacy could be remedied by an additional test comparing the distances of the query point from the two competing sites. Although effective, such ad hoc solution would not fit the exact-computation paradigm, whereas our method fully complies with it. □

4.4. Implicit higher-order Voronoi diagrams. In this section, we introduce implicit higher-order Voronoi diagrams for point sites in the plane, and we extend the results of section 4.3 to k-nearest neighbors and circular range search queries.

The definition of the *implicit order-k Voronoi diagram* $V_k^*(S)$ of set S of point sites in the plane is analogous to that given in section 4.3 for Voronoi diagrams. A vertex v of $V_k(S)$ is represented by its approximate coordinates $x^*(v)$ and $y^*(v)$, and a nonhorizontal edge e of $V_k(S)$ 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 $r(e)$.

Lemmas 10–11 immediately hold also for $V_k(S)$, and we obtain Lemma 14.

LEMMA 14. *For any native method for monotone maps, a point-location query in an implicit order-k Voronoi diagram has optimal degree* 2 *and has the same asymptotic time complexity as a point location query in an explicit order-k Voronoi diagram.*

The order-k Voronoi diagram $V_k(S)$ for a set S of n point sites has $O(k(n-k))$ vertices, edges, and faces and can be obtained from the order $k-1$ implicit Voronoi diagram $V_{k-1}(S)$ by intersecting each face of $V_{k-1}(S)$ with the (order-1) Voronoi diagram of a suitable subset of the vertices of S [43]. As shown in [43, 16], $V_k(S)$ can be computed in $O(k(n-k)\sqrt{n}\log n)$ time. Since the construction is based on iteratively computing Voronoi diagrams by using the incircle test, which is the most expensive operation in terms of degree, the overall degree of the preprocessing is 4 (Lemma 5). Hence, we obtain Lemma 15.

LEMMA 15. *The implicit order-k Voronoi diagram of n point sites in the plane can be computed in $O(k(n-k)\sqrt{n}\log n)$ time, $O(k(n-k))$ space, and with degree* 4.

Point location in the order-k Voronoi diagram solves k-nearest neighbors queries. Hence, by Theorem 3 and Lemmas 14–15, we obtain Theorem 7.

THEOREM 7. *Let S be a set of n point sites in the plane and k an integer with $1 \le k \le n-1$. 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.

Circular range search queries in a set S of n point sites can be reduced to a sequence of 2^i-nearest neighbors queries in $V_{2^i}(S)$, $i = 0, \ldots, \log n$ [7]. This approach yields a data structure with $O(n^3)$ space and preprocessing time, and $O(\log n \log\log n + k)$ query time, where k is the size of the output. Hence, with analogous reasoning as above, we obtain the following theorem.

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.

The space and preprocessing time of Theorems 7–8 and the query time of Theorem 8 can be improved while preserving the same degree bounds by more complicated procedures along the lines suggested in [1, 2, 15].

5. Proximity queries for point sites in 3D space. In this section, we consider the following proximity query on a set S of point sites in 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 2D case, such a query is efficiently answered by performing point location in the 3D Voronoi diagram of S. Test primitives and methods for spatial point location are described in section 5.1. Section 5.2 shows that "conventional" algorithms require degree 8. A degree-3 algorithm based on "implicit" 3D Voronoi diagrams is then given in section 5.3.

5.1. Test primitives and methods for spatial point location. There are only two known efficient spatial point location methods for cell-complexes that are applicable to 3D Voronoi diagrams: the *separating surfaces method* [14, 56], which extends the chain method [44], and the *persistent planar location method* [52], which extends the persistent search tree method [53]. 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.

As in section 4.1, 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) compares the z-coordinate of the query point q with the z-coordinate of vertex v.

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

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

left-right(q_{xy}, e_{xy}) compares 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) determines 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).

Now, we analyze the separating surfaces method for spatial point location [14, 56] in acyclic cell-complexes. *Separating surfaces* are the 3D analogue of separators of monotone maps. Their existence is guaranteed by the acyclicity of the cell-complex. Thus, a point location query consists of traversing a root-to-leaf path in the *separating surface tree*, where at each node we determine whether the query point q is to above or below the separating surface associated with the node. The discrimination of point q with respect to a separating σ is performed in two steps:

1. By means of a planar point location query for the xy-projection q_{xy} of q in the xy projection of σ, we find the facet f of σ whose xy projection contains q_{xy}. If an ordinary point location method is used, this step uses primitives front-rear(q, v), left-right(q, v), and left-right(q_{xy}, e_{xy}).
2. We discriminate q with respect to σ by performing test above-below(q, f).

In the special cases that cell-complex C has facets parallel to the z-axis, the discrimination of point q with respect to σ uses also test primitives above-below(q, v). Thus, the separating surfaces method is native for acyclic convex cell-complexes.

A similar analysis shows that also the persistent planar location method is native for convex cell-complexes. More specifically, we have Lemma 16.

LEMMA 16. *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.

5.2. Explicit 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 17. *The* left-right(q_{xy}, e_{xy}) *test primitive in an explicit Voronoi diagram of point sites in* 3D *space has degree* 8.

Proof. The reasoning is the same as in the proof of Lemma 9. Let $e_{x,y} \equiv (v_1, v_2)$, where v_1 and v_2 are the xy-projections of two adjacent vertices u and v of $V(S)$; let u be equidistant from the four primitive sites $a \equiv (x_a, y_a)$, $b \equiv (x_b, y_b)$, $c \equiv (x_c, y_c)$, and

$d \equiv (x_d, y_d)$, and v from $a \equiv (x_a, y_a)$, $b \equiv (x_b, y_b)$, $c \equiv (x_c, y_c)$, and $h \equiv (x_h, y_h)$. In an explicit Voronoi diagram, test primitive left-right(q_{xy}, e_{xy}) that determines whether query point $q \equiv (x_q, y_q)$ is to the left or to the right of edge $e \equiv (v_1, v_2)$ is equivalent to evaluating the sign of the following determinant:

$$\Delta = \begin{vmatrix} x_q & y_q & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = \begin{vmatrix} x_q & y_q & 1 \\ \frac{X_1}{2W_1} & \frac{Y_1}{2W_1} & 1 \\ \frac{X_2}{2W_2} & \frac{Y_2}{2W_2} & 1 \end{vmatrix} = \frac{1}{4W_1W_2} \begin{vmatrix} x_q & y_q & 1 \\ X_1 & Y_1 & 2W_1 \\ X_2 & Y_2 & 2W_2 \end{vmatrix} = \frac{\Delta'}{4W_1W_2},$$

where

$$X_1 = \begin{vmatrix} x_a^2 + y_a^2 + z_a^2 & y_a & z_a & 1 \\ x_b^2 + y_b^2 + z_b^2 & y_b & z_b & 1 \\ x_c^2 + y_c^2 + z_c^2 & y_c & z_c & 1 \\ x_d^2 + y_d^2 + z_d^2 & y_d & z_d & 1 \end{vmatrix}, \quad Y_1 = \begin{vmatrix} x_a & x_a^2 + y_a^2 + z_a^2 & z_a & 1 \\ x_b & x_b^2 + y_b^2 + z_b^2 & z_b & 1 \\ x_c & x_c^2 + y_c^2 + z_c^2 & z_c & 1 \\ x_d & x_d^2 + y_d^2 + z_d^2 & z_d & 1 \end{vmatrix},$$

$$W_1 = \begin{vmatrix} x_a & y_a & z_a & 1 \\ x_b & y_b & z_b & 1 \\ x_c & y_c & z_c & 1 \\ x_d & y_d & z_d & 1 \end{vmatrix},$$

and X_2, Y_2, and W_2 have similar expressions obtained replacing in the above determinants x_d with x_h, y_d with y_h, and z_d with z_h.

Evaluating the sign of Δ is equivalent to evaluating the signs of W_1, W_2 and of Δ'.

By using the notation introduced in section 2, we can rewrite X_i and Y_i as α^4, and W_i as α^3 $(i = 1, 2)$. Hence, Δ' is a degree-8 multivariate polynomial since it can be rewritten as

$$\alpha(\alpha^4\alpha^3 - \alpha^4\alpha^3) - \alpha(\alpha^4\alpha^3 - \alpha^4\alpha^3) + \alpha^4\alpha^3 - \alpha^4\alpha^4 \quad \longrightarrow^{(2,3,4)} \quad \alpha^8 + \alpha^8 \quad \longrightarrow^{(3)} \alpha^8. \qquad \square$$

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 17, we have Theorem 9.

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

5.3. Implicit Voronoi diagrams. 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 2D Voronoi diagrams given in section 4.3. Namely, $V^*(S)$ stores the topological structure of the 3D Voronoi diagram $V(S)$ of S (e.g., the data structure of [23]) 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 2D case (see section 4.3), it is easy to see that such implementations are correct.

Lemma 18. *Test primitives* above-below(q, v)*,* left-right(q, v)*,* front-rear(q, v) *in an implicit Voronoi diagram of* 3D *point sites can be performed in* $O(1)$ *time and with degree* 1*.*

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 4, we have Lemma 19.

Lemma 19. *Test primitive* above-below(q, f) *in an implicit Voronoi diagram of* 3D *point sites 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 20. *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*.*

Proof. The line containing the oriented edge e_{xy} is the projection on the xy-plane of the intersection of two planes containing two facets of the 3D Voronoi diagram. Let $a_i x + b_i y + c_i z + d_i = 0$ be the equation of one such plane ($i = 1, 2$). The projection of their intersection on the xy-plane is

$$\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix} x + \begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix} y + \begin{vmatrix} d_1 & c_1 \\ d_2 & c_2 \end{vmatrix} = 0.$$

Test left-right(q_{xy}, e_{xy}) consists of determining the sign of

$$\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix} x_q + \begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix} y_q + \begin{vmatrix} d_1 & c_1 \\ d_2 & c_2 \end{vmatrix},$$

which is a multivariate polynomial having arithmetic degree 3, since it can be rewritten as

$$\alpha\alpha^2 + \alpha\alpha^2 + \alpha^3 \longrightarrow^{(2,3)} \alpha^3. \qquad \square$$

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 18–20 we obtain Lemma 21.

Lemma 21. *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.

The implicit Voronoi diagram $V^*(S)$ of a set S of n points in 3D space can be constructed by computing the 3D Delaunay triangulation with the incremental algorithm by Joe [40], whose time complexity and storage is $O(n^2)$ (see also [49]). Since the most demanding operation of the algorithm in terms of degree is the 3D insphere test, from Lemma 6 we have that the degree of the algorithm that computes $V(S)$ is 5. As in the planar case, the topological structure of $V(S)$ and the sites $\ell(f)$ and $r(f)$ for each edge e of $V(S)$ are immediately derived from $DT(S)$ by duality. We then compute the approximations $x^*(v)$, $y^*(v)$, and $z^*(v)$ for each vertex v of $V(S)$ by means of integer division. Let a, b, c, and d be the four sites of S that define vertex v. Adopting the same notation as in the proof of Lemma 17, the x-coordinate $x(v)$ of v is given by the ratio $x(v) = \frac{Y_1}{2W_1}$, where X_1 is a variable of arithmetic degree 4 and W_1 is a variable of arithmetic degree 3; this is similar for $y(v)$ and $z(v)$. We summarize the above analysis as follows.

LEMMA 22. *The implicit Voronoi diagram of a set of n point sites in* 3D *space can be computed in $O(n^2)$ time and space and with degree* 5.

Lemmas 21 and 22 lead to the following theorem.

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.

Proof. We perform point location in the implicit Voronoi $V^*(S)$ diagram of S using the separating surfaces method. The space requirement and the query degree and time follow from the performance of these methods and from Lemma 21.

Regarding the preprocessing time, by Lemma 22, the construction of the implicit Voronoi $V^*(S)$ takes $O(n^2)$ time with degree 5. In order to construct the point-location data structure, we also need an additional test primitive that consists of comparing the y-coordinates of two Voronoi vertices. For example, this primitive is used to establish bridges between the vertices of the different separating chains if the bridged-chain method (see section 4.1) is applied to locate the xy-projection of the query point into the xy-projection of a separating surface. Comparing the y-coordinates of the Voronoi vertices is equivalent to evaluating the sign of multivariate polynomials of the form $\frac{Y_i}{2W_i} - \frac{Y_j}{2W_j}$, where $\frac{Y_i}{2W_i}$ and $\frac{Y_j}{2W_j}$ represent the y-coordinates of two different Voronoi vertices (see also the proof of Lemma 17). Such multivariate polynomials have degree 7, since they can be rewritten as

$$\frac{\rho_i}{\rho_j} - \frac{\rho_h}{\rho_k} \quad \longrightarrow^{(6)} \quad \rho_i\rho_k - \rho_h\rho_j \quad \longrightarrow^{(1)} \alpha^4\alpha^3 - \alpha^4\alpha^3 \quad \longrightarrow^{(2,3,4)} \quad \alpha^7. \qquad \square$$

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.

6. Proximity queries for 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 a query is efficiently solved by performing point location in the Voronoi diagram of the set of point and segment sites [51].

The test primitives needed by such an approach are described in section 6.1. Section 6.2 shows that the "conventional" approach requires degree 64. A degree-6 algorithm based on "implicit" Voronoi diagrams is then given in section 6.3.

6.1. Test primitives and methods. 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

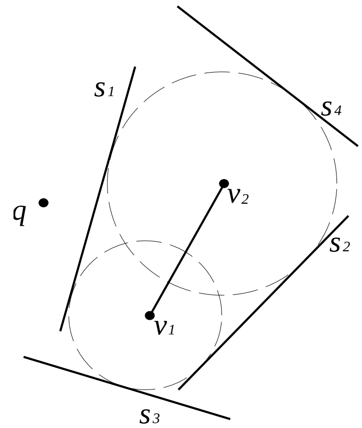

FIG. 2. *Illustration for Lemma* 24.

those defined in section 4.1 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 23. *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.*

6.2. Explicit Voronoi diagrams. 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 [10, 59].

In the following lemma, we analyze the degree of test primitive $\mathsf{left\text{-}right}(q, e)$ for a straight-line edge e of an explicit extended Voronoi diagram.

LEMMA 24. *The* $\mathsf{left\text{-}right}(q, e)$ *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.

Proof. Let $e \equiv (v_1, v_2)$, such that $v_1 \equiv (x_1, y_1)$ is equidistant from three segments s_1, s_2, and s_3 and v_2 is from three segments s_1, s_2, and s_4. See Figure 2.

We show that the test $\mathsf{left\text{-}right}(q, e)$ for determining whether the query point $q \equiv (x_q, y_q)$ is to the left or to the right of (v_1, v_2) has degree 64. Namely, let $a_i x + b_i y + c_i = 0$ $(i = 1, 2, 3, 4)$ be the equation of the line containing segment s_i. In an explicit Voronoi diagram, test primitive $\mathsf{left\text{-}right}(q, e)$, determines whether query point $q \equiv (x_q, y_q)$ is to the left or to the right of edge $e \equiv (v_1, v_2)$, is equivalent to evaluating the sign of the following determinant:

$$\Delta = \begin{vmatrix} x_q & y_q & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = \begin{vmatrix} x_q & y_q & 1 \\ \frac{X_1}{W_1} & \frac{Y_1}{W_1} & 1 \\ \frac{X_2}{W_2} & \frac{Y_2}{W_2} & 1 \end{vmatrix} = \frac{1}{W_1 W_2} \begin{vmatrix} x_q & y_q & 1 \\ X_1 & Y_1 & W_1 \\ X_2 & Y_2 & W_2 \end{vmatrix} = \frac{\Delta'}{W_1 W_2},$$

where

$$X_1 = \begin{vmatrix} b_1 & c_1 & \sqrt{a_1^2 + b_1^2} \\ b_2 & c_2 & \sqrt{a_2^2 + b_2^2} \\ b_3 & c_3 & \sqrt{a_3^2 + b_3^2} \end{vmatrix}, \quad Y_1 = \begin{vmatrix} a_1 & c_1 & \sqrt{a_1^2 + b_1^2} \\ a_2 & c_2 & \sqrt{a_2^2 + b_2^2} \\ a_3 & c_3 & \sqrt{a_3^2 + b_3^2} \end{vmatrix},$$

$$W_1 = \begin{vmatrix} b_1 & a_1 & \sqrt{a_1^2 + b_1^2} \\ b_2 & a_2 & \sqrt{a_2^2 + b_2^2} \\ b_3 & a_3 & \sqrt{a_3^2 + b_3^2} \end{vmatrix},$$

and X_2, Y_2, and W_2 have similar expressions obtained by substituting in the above determinants a_3 with a_4, b_3 with b_4, and c_3 with c_4. Evaluating the sign of Δ is equivalent to evaluating the signs of W_1, W_2 and of Δ'. In the rest of this proof we show that evaluating the sign of Δ' is a computation with degree 64. By using the same technique, one can easily see that evaluating the signs of W_1 and W_2 is a computation with degree 12.

We have

$$\Delta' = x_q(Y_2W_1 - Y_1W_2) - y_q(X_1W_2 - X_2W_1) + (X_2Y_1 - X_1Y_2). \tag{1}$$

By using the notation introduced in section 2, we can rewrite X_1, and Y_1 as $\alpha^3\rho_1 + \alpha^3\rho_2 + \alpha^3\rho_3$, W_1 as $\alpha^2\rho_1 + \alpha^2\rho_2 + \alpha^2\rho_3$, X_2 and Y_2 as $\alpha^3\rho_1 + \alpha^3\rho_2 + \alpha^3\rho_4$, and W_2 as $\alpha^2\rho_1 + \alpha^2\rho_2 + \alpha^2\rho_4$, where $\rho_i = \sqrt{a_i^2 + b_i^2}$ $(i = 1, \dots, 4)$. Considering that x_q and y_q are expressions of type α and applying repeatedly Rules (1) and (2), we obtain the expression

$$\alpha^8 + \alpha^6\rho_1\rho_2 + \alpha^6\rho_1\rho_3 + \alpha^6\rho_1\rho_4 + \alpha^6\rho_2\rho_3 + \alpha^6\rho_2\rho_4 + \alpha^6\rho_3\rho_4.$$

By means of the rewriting rules of section 2 we have

$$\begin{array}{ll}
\alpha^8 + \alpha^6\rho_1\rho_2 + \alpha^6\rho_1\rho_3 + \alpha^6\rho_1\rho_4 + \alpha^6\rho_2\rho_3 + \alpha^6\rho_2\rho_4 + \alpha^6\rho_3\rho_4 & \longrightarrow^{(7)} \\
(\alpha^8 + \alpha^6\rho_2\rho_3 + \alpha^6\rho_2\rho_4 + \alpha^6\rho_3\rho_4)^2 - (\alpha^6\rho_1\rho_2 + \alpha^6\rho_1\rho_3 + \alpha^6\rho_1\rho_4)^2 & \longrightarrow^{(1,2,3,4)} \\
\alpha^{16} + \alpha^{14}\rho_2\rho_3 + \alpha^{14}\rho_2\rho_4 + \alpha^{14}\rho_3\rho_4 & \longrightarrow^{(7)} \\
(\alpha^{16} + \alpha^{14}\rho_2\rho_3)^2 - (\alpha^{14}\rho_2\rho_4 + \alpha^{14}\rho_3\rho_4)^2 & \longrightarrow^{(1,2,3,4)} \\
\alpha^{32} + \alpha^{30}\rho_2\rho_3 & \longrightarrow^{(7)} \\
\alpha^{64} - \alpha^{64} & \longrightarrow^{(3,4)} \\
\alpha^{64}. &
\end{array}$$

$\square$

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 24 we conclude Theorem 11.

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.

6.3. Implicit Voronoi diagrams. 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 4.3. Namely, $V^*(S)$ stores the topological structure of the extended Voronoi diagram $V'(S)$ of S (e.g., the data structure of [23]) 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 nonhorizontal 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 Lemmas 2–4, we have Lemma 25.

LEMMA 25. *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.*

The implicit Voronoi diagram can be constructed in $O(n \log n)$ expected running time by using the randomized incremental algorithm of [11]. The most demanding operation is incircle test for three segments, which has degree 40 by Lemma 7 (see also [9]). By a similar analysis as the one shown in sections 4 and 5, it is not hard to show that both the y-ordering of the vertices of $V(S)$ and the semi-integer approximation of the vertex coordinates can be performed without affecting the computational cost and the degree of the computation of $V(S)$.

LEMMA 26. *The implicit Voronoi diagram of a set of n point and segment sites in the plane can be computed in $O(n \log n)$ expected time, $O(n)$ space, and degree* 40*.*

Lemmas 25 and 26 lead to the following theorem.

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*.*

7. 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 3D 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 4.3.

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 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 nonhorizontal 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.

The following lemmas extend Lemmas 12–13 and can be proved with similar techniques.

LEMMA 27. *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 28. *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.

By the above discussion and Lemmas 27–28, we obtain the following theorem that improves upon Theorem 6.

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 3D space. This reduces the degree of the preprocessing from 5 to 4 in Theorems 7–8 and from 7 to 5 in Theorem 10.

8. Further research directions. Within the proposed approach, this paper only addresses the issue of the degree of test computations and illustrates its impact on algorithmic design in relation to a central problem in computational geometry. However, several important related problems need further investigation and will be reported on in the near future.

First, the methodological framework described in section 2 should be extended to the computation of the degree of other classes of geometric primitives. Recently, motivated in part by a preliminary version of this paper [45], Burnikel et al. [13] have presented a new separation bound for arithmetic expressions involving square roots.

Also, since the degree of an algorithm expresses worst-case computational requirement occurring in degenerate or near-degenerate instances, special attention must be devoted to the development of a methodology that reliably computes the sign of an expression with the least expenditure of computational resources. This involves the use of "arithmetic filters," possibly families of filters, of progressively increasing power that, depending upon the values of primitive variables, carefully adjust the computational effort (see, e.g., [4, 11, 29, 41]).

Next, one should carefully analyze the precision adopted in executing constructions, so that the outputs are within the precision bounds required by the application. In addition, each construction algorithm should be accompanied by a verification algorithm, which not only checks for topological compliance of the output with the generic member of its class (as, e.g., a Voronoi diagram must have the topology of a convex map) as illustrated in [54] but more specifically verifies its topological agreement with the structure emerging from the tests executed by the algorithm [47].

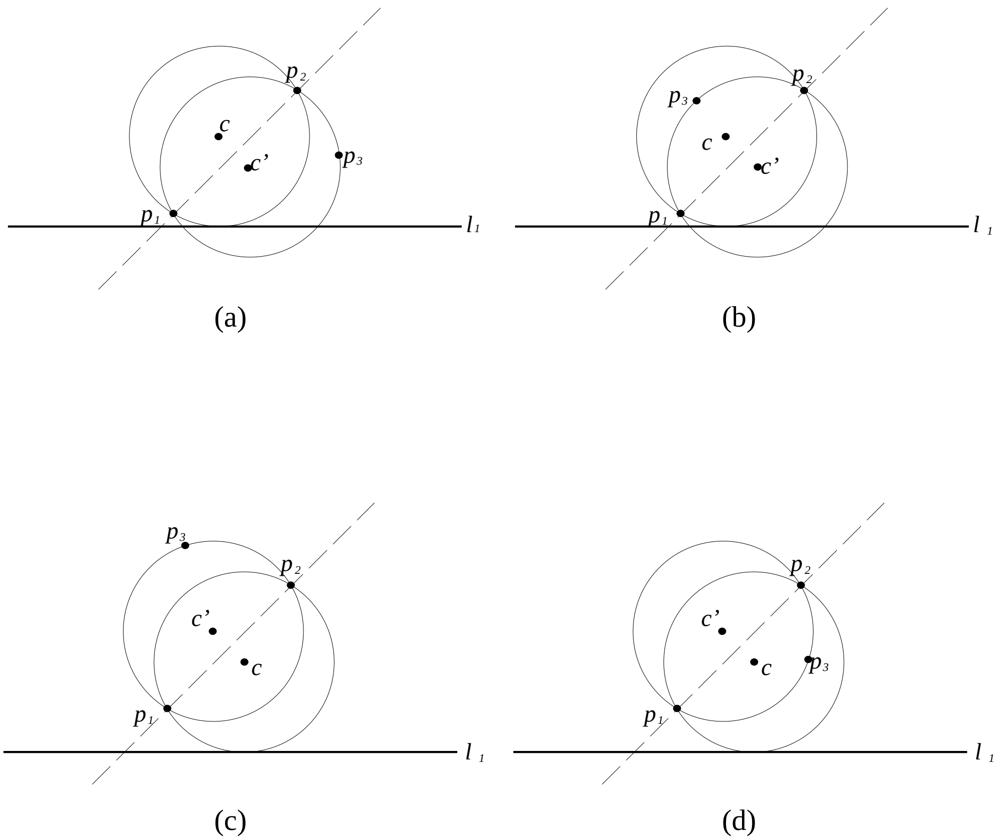

FIG. 3. *Different cases for test* $(p_1, p_2, l_1; p_3)$.

Beyond these general methodological issues, the investigation reported in these pages leaves some interesting open problems such as answering nearest neighbor queries in subquadratic time and optimal degree for a set of points in 3D space or improving the efficiency of the preprocessing stage in computing the implicit Voronoi diagram of a set of sites.

We mention, in this respect, how the degree can impact the design of geometric primitives adopted in existing algorithms for Voronoi diagrams of point and segment sites. Let $(a_1, a_2, a_3; a_4)$, with a_i either a point or a segment, denote the incircle test, where a_4 is tested for intersection with circle(a_1, a_2, a_3). Specifically, consider $(p_1, p_2, l_1; p_3)$, which can be answered with degree 12 [9]. We show that it can be more efficiently executed as follows. First perform the test $(p_1, p_2, p_3; l_1)$. Let c and c' be the centers of circle(p_1, p_2, l_1) and circle(p_1, p_2, p_3), respectively. Two cases are possible: either circle(p_1, p_2, p_3) intersects l_1 or it does not. In the first case, p_3 is inside circle(p_1, p_2, l_1) if and only if c' and p_3 lie on opposite sides of line $\overline{p_1 p_2}$ through p_1 and p_2 (see Figures 3(a) and 3(b)). In the second case the answer to test $(p_1, p_2, l_1; p_3)$ depends on which side of $\overline{p_1 p_2}$ point p_3 lies (see Figures 3(c) and 3(d)). Thus, test $(p_1, p_2, l_1; p_3)$ is reduced to test $(p_1, p_2, p_3; l_1)$ that can be executed with degree 8 (see [9]) and at most two other left-right tests of lower degree.

Finally, an important issue for future research deals with the experimental comparison between point location algorithms in implicit Voronoi diagrams and traditional point location algorithms in explicit Voronoi diagrams. We are currently implementing *GeomLib*, an object-oriented library for robust geometric computing that will be accessible through the world wide web by using the architectural framework of *Mocha* [6, 5].

Acknowledgment. The authors wish to thank the referees for several useful suggestions.

REFERENCES

[1] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor, *A linear-time algorithm for computing the Voronoi diagram of a convex polygon*, Discrete Comput. Geom., 4 (1989), pp. 591–604.

[2] A. Aggarwal, M. Hansen, and T. Leighton, *Solving query-retrieval problems by compacting Voronoi diagrams*, in Proc. 22nd Annu. ACM Sympos. Theory Comput., 1990, Association for Computing Machinery, New York, pp. 331–340.

[3] F. Aurenhammer, *Voronoi diagrams: A survey of a fundamental geometric data structure*, ACM Comput. Surv., 23 (1991), pp. 345–405.

[4] 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.

[5] J. E. Baker, I. F. Cruz, G. Liotta, and R. Tamassia, *The Mocha algorithm animation system*, ACM Comput. Surv., 27 (1995), pp. 568–572.

[6] J. E. Baker, I. F. Cruz, G. Liotta, and R. Tamassia, *Animating geometric algorithms over the Web*, in Proc. 12th Annu. ACM Sympos. Comput. Geom., Association for Computing Machinery, New York, 1996, pp. C3–C4.

[7] J. L. Bentley and H. A. Maurer, *A note on Euclidean near neighbor searching in the plane*, Inform. Process. Lett., 8 (1979), pp. 133–136.

[8] M. Bocher, *Introduction to Higher Algebra*, Macmillan, New York, 1907.

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

[10] 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., Association for Computing Machinery, New York, 1995, pp. C18–C19.

[11] C. Burnikel, K. Mehlhorn, and S. Schirra, *How to compute the Voronoi diagram of line segments: Theoretical and experimental results*, in 2nd Annual European Symp. on Algorithms, Lecture Notes Comput. Sci. 855, Springer-Verlag, Berlin, 1994, pp. 227–239.

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

[13] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra, *A strong and easily computable separation bound for arithmetic expressions involving square roots*, in Proc. ACM-SIAM Symposium on Discrete Algorithms, 1997.

[14] B. Chazelle, *How to search in history*, Inform. Control, 64 (1985), pp. 77–99.

[15] B. Chazelle, R. Cole, F. P. Preparata, and C. K. Yap, *New upper bounds for neighbor searching*, Inform. Control, 68 (1986), pp. 105–124.

[16] B. Chazelle and H. Edelsbrunner, *An improved algorithm for constructing kth-order Voronoi diagrams*, IEEE Trans. Comput., C-36 (1987), pp. 1349–1354.

[17] B. Chazelle and L. J. Guibas, *Fractional cascading:* I. *A data structuring technique*, Algorithmica, 1 (1986), pp. 133–162.

[18] B. Chazelle and L. J. Guibas, *Fractional cascading:* II. *Applications*, Algorithmica, 1 (1986), pp. 163–191.

[19] K. L. Clarkson, *Safe and effective determinant evaluation*, in Proc. 33rd Ann. IEEE Sympos. Found. Comput. Sci., IEEE Press, Piscataway, NJ, 1992, pp. 387–395.

[20] T. K. Dey, K. Sugihara, and C. L. Bajaj, *Delaunay triangulations in three dimensions with finite precision arithmetic*, Comput. Aided Geom. Design, 9 (1992), pp. 457–470.

[21] D. P. Dobkin, *Computational geometry and computer graphics*, in Proc. IEEE, 80 (1992), pp. 1400–1411.

[22] D. P. Dobkin and D. G. Kirkpatrick, *Fast detection of polyhedral intersection*, Theoret. Comput. Sci., 27 (1982), pp. 241–253.

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

[24] M. Edahiro, I. Kokubo, and T. Asano, *A new point-location algorithm and its practical efficiency: Comparison with existing algorithms*, ACM Trans. Graph., 3 (1984), pp. 6–109.

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

[26] 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 (1985), pp. 39–47.

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

[28] S. Fortune, *Stable maintenance of point set triangulations in two dimensions*, in Proc. 30th Ann. IEEE Sympos. Found. Comput. Sci., IEEE Press, Piscataway, NJ, 1989, pp. 494–505.

[29] S. Fortune, *Numerical stability of algorithms for* 2*-d Delaunay triangulations*, Internat. J. Comput. Geom. Appl., 5 (1995), pp. 193–213.

[30] S. Fortune, *Polyhedral modeling with multiprecision integer arithmetic*, Comput. Aided Design, to appear.

[31] S. Fortune and C. J. Van Wyk, *Efficient exact arithmetic for computational geometry*, in Proc. 9th Annu. ACM Sympos. Comput. Geom., Association for Computing Machinery, New York, 1993, pp. 163–172.

[32] S. Fortune and C. V. Wyk, *Static analysis yields efficient exact integer arithmetic for computational geometry*, ACM Trans. Graphics, 15 (1996), pp. 223–248.

[33] S. J. Fortune, *A sweepline algorithm for Voronoi diagrams*, Algorithmica, 2 (1987), pp. 153–174.

[34] D. H. Greene and F. F. Yao, *Finite-resolution computational geometry*, in Proc. 27th Ann. IEEE Sympos. Found. Comput. Sci., IEEE Press, Piscataway, NJ, 1986, pp. 143–152.

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

[36] L. J. Guibas, D. Salesin, and J. Stolfi, *Epsilon geometry: Building robust algorithms from imprecise computations*, in Proc. 5th Ann. ACM Sympos. Comput. Geom., Association for Computing Machinery, New York, 1989, pp. 208–217.

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

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

[39] C. M. Hoffmann, J. E. Hopcroft, and M. T. Karasick, *Robust set operations on polyhedral solids*, IEEE Comput. Graph. Appl., 9 (1989), pp. 50–59.

[40] B. Joe, *Construction of three-dimensional Delaunay triangulations using local transformations*, Comput. Aided Geom. Design, 8 (1991), pp. 123–142.

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

[42] D. G. Kirkpatrick, *Optimal search in planar subdivisions*, SIAM J. Comput., 12 (1983), pp. 28–35.

[43] D. T. Lee, *On k-nearest neighbor Voronoi diagrams in the plane*, IEEE Trans. Comput., C-31 (1982), pp. 478–487.

[44] D. T. Lee and F. P. Preparata, *Location of a point in a planar subdivision and its applications*, SIAM J. Comput., 6 (1997), pp. 594–606.

[45] G. Liotta, F. P. Preparata, and R. Tamassia, *Robust Proximity Queries in Implicit Voronoi Diagrams*, Technical Report CS-96-16, Center for Geometric Computing, Comput. Sci. Dept., Brown Univ., Providence, RI, 1996.

[46] K. Mehlhorn and S. Näher, *LEDA: A platform for combinatorial and geometric computing*, Comm. ACM, 38 (1995), pp. 96–102.

[47] 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 Ann. ACM Sympos. Comput. Geom., Association for Computing Machinery, New York, 1996, pp. 159–165.

[48] V. J. Milenkovic, *Verifiable implementations of geometric algorithms using finite precision arithmetic*, Artif. Intell., 37 (1988), pp. 377–401.

[49] E. Mücke, *Detri* 2.2: *A robust implementation for* 3d *Triangulations*, manuscript, http://www.geom.umn.edu:80/software/cglist/lowdvod.html (1996).

[50] F. P. Preparata, *A new approach to planar point location*, SIAM J. Comput., 10 (1981), pp. 473–482.

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

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

[53] N. Sarnak and R. E. Tarjan, *Planar point location using persistent search trees*, Comm. ACM, 29 (1986), pp. 669–679.

[54] K. Sugihara and M. Iri, *Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic*, Proc. IEEE, IEEE Press, Piscataway, NJ, 80 (1992), pp. 1471–1484.

[55] K. Sugihara, Y. Ooishi, and T. Imai, *Topology-oriented approach to robustness and its applications to several Voronoi-diagram algorithms*, in Proc. 2nd Canad. Conf. Comput. Geom., 1990, pp. 36–39.
[56] R. Tamassia and J. S. Vitter, *Optimal cooperative search in fractional cascaded data structures*, Algorithmica, 15 (1996), pp. 154–171.
[57] C. Yap and T. Dubé, *A Basis for Implementing Exact Geometric Algorithms*, manuscript, http://simulation.nyu.edu/projects/exact/references.html (1993).
[58] C. K. Yap, *Symbolic treatment of geometric degeneracies*, J. Symbolic Comput., 10 (1990), pp. 349–370.
[59] C. K. Yap, *Toward exact geometric computation*, Comput. Geom., 7 (1997), pp. 3–23.