An Efficient Approach to Removing Geometric Degeneracies

(Extended Abstract)

Ioannis Emiris *
emiris@cs.berkeley.edu

John Canny*
jfc@cs.berkeley.edu

Computer Science Division
University of California, Berkeley

Abstract

Our aim is to perturb the input so that an algorithm designed under the hypothesis of input non-degeneracy can execute on arbitrary instances. The deterministic scheme of [EmCa] was the first efficient method and was applied to two important predicates. Here it is extended in a consistent manner to another two common predicates, thus making it valid for most algorithms in computational geometry. It is shown that this scheme incurs no extra algebraic complexity over the original algorithm while it increases the bit complexity by a factor roughly proportional to the dimension of the geometric space. The second contribution of this paper is a variant scheme for a restricted class of algorithms that is asymptotically optimal with respect to the algebraic as well as the bit complexity. Both methods are simple to implement and require no symbolic computation. They also conform to certain criteria ensuring that the solution to the original input can be restored from the output on the perturbed input. This is immediate when the input to solution mapping obeys a continuity property and requires some case-specific work otherwise. Finally we discuss extensions and limitations to our approach.

1 Introduction

Algorithms in computational geometry are typically designed for input instances in general position. The treatment of degenerate cases is tedious and intricate, thus seldom included in the theoretical discussion, yet it remains a nontrivial matter for implementors. In this paper we describe a general approach to eliminate the need of dealing with degeneracies.

Take, for instance, the Convex Hull problem in d dimensions and consider your favorite algorithm for it. If defined under the hypothesis of non-degeneracy, the algorithm assumes that no more than d points lie on the same $(d-1)$-dimensional hyperplane. This supposes that some relevant predicate, called Orientation below, can always decide the side on which the $(d+1)$-st point lies with respect to the hyperplane spanned by the first d points. This assumption does not always hold in practice. Systematic methods that have been proposed introduce an infinitesimal perturbation of the input points and this is also our approach. For instance, perturbing 4 coplanar points in 3-dimensional space will result in one of the following two configurations. Either one face defined by exactly 3 of the points with the fourth one in the interior of the convex hull, or all points on the hull defining two or three adjacent faces. Both scenarios will produce a convex hull arbitrarily close to, if not the same as, that of the unperturbed point set.

Previous approaches include [EdMu] and [Ya87]; their main drawback is that they incur a time complexity at least exponential in the space dimension. The direct deterministic perturbation of Emiris and Canny [EmCa] is the first efficient method in the sense that the complexity overhead caused by introducing the perturbation is a small *constant* factor. Counting bit operations, the overhead is a *polynomial* in the dimension d, namely $\mathcal{O}(d^{1+\alpha})$, for any positive constant α arbitrarily close to zero. It was applied to predicates Orientation and Transversality and here, following the presentation in [Em], is extended to Ordering and InSphere thus covering most geometric algorithms. The overhead is as above for the InSphere and constant for the Ordering predicate.

Secondly, we define a variant suggested by Seidel [Se] which is optimal in the bit complexity and prove that it applies to Orientation and Transversality. This scheme

*Supported by a David and Lucile Packard Foundation Fellowship and by NSF Presidential Young Investigator Grant IRI-8958577.

leaves both algebraic and bit complexities unaffected In addition to their efficiency, our schemes are simple to implement and require no symbolic computation, despite the presence of a symbolic infinitesimal variable in the definition.

Neither scheme affects the output on non-degenerate input; moreover, they both produce a consistent approximate solution, arbitrarily close to that of the unperturbed input provided that the latter exists. We capture these properties in the definition of a *valid* perturbation.

The next section provides all definitions. Section 3 is a comparative study of previous work on handling degeneracies and Section 4 defines our perturbation schemes, along with some other alternatives. The two main schemes are shown to be valid for the four predicates of Section 5, where the complexity claims are also proven. This section also refers to extensions and limitations of our technique. We conclude with a summary of the main results and future directions in our work.

2 Preliminaries

The issues of this section are elaborated upon in [EmCa]; here we discuss briefly these points that are indispensable in subsequent sections.

2.1 Computational model

We choose the real RAM [PrSh], and also make reference to a *program* for it, which is a sequence of instructions that implements a specific algorithm. The *input* to the RAM, or to a particular program, is a set of n vectors in $\mathbf{R}^d$, where $\mathbf{R}$ denotes the set of real numbers, n and d are positive integers, and $n > d$. Let $p_{i,j}$ denote the j-th coordinate of the i-th vector, with $1 \leq i \leq n$ and $1 \leq j \leq d$. Only the four basic operations $\{+, -, \times, /\}$ are allowed between real numbers and are assumed to be exact, where the operands are constants, input parameters or have been computed by the program previously.

Branching occurs at tests of an input or a computed quantity against zero and is 3-way, depending on the sign of the tested value. We require that the tested quantity be expressible as the value of a polynomial in the input variables which, although excluding certain tests on new derived objects, allows the model to capture most important algorithms. A further restriction in this paper is to concentrate on only four tests, called Ordering, Orientation, Transversality and InSphere, each expressed as a determinant of some matrix in the input parameters; they are explicitly given in Section 5.

We make use of two complexity measures. Under the *algebraic model* the total cost equals the number of vertices in the longest computation path. More realistically, we must keep track of the operands' bit size. Under the *bit model* only the input, output and branching vertices are assigned unit cost. The latter implies a property of the representation; however, even if zero-testing takes time proportional to the size of the operand all of our results hold. For integers of size $\mathcal{O}(b)$, addition and subtraction require $\mathcal{O}(b)$ while multiplication and division require $M(b)$ bit operations, where $M(b) = \mathcal{O}(b \log b \log\log b)$, [AHU]; the latter is the upper bound on the bit complexity of any arithmetic operation between any two numbers in the RAM. Then the total bit complexity equals the sum of the costs of every vertex on a computation path, maximized over all paths.

2.2 Degeneracy

We distinguish two kinds of degeneracies. *Intrinsic* ones depend solely on the problem at hand, such as having more than 2 collinear points in the 2-dimensional Convex Hull problem. On the other hand, covertical points have no effect on the convex hull itself, but may complicate a sweep-line algorithm. This is an instance of an *algorithm-dependent* degeneracy. We focus on the latter because they subsume intrinsic degeneracies.

Definition 2.1 *An input instance* $\mathbf{p}$ *is* degenerate *with respect to some algorithm if it causes the determinant* f *at some branching vertex of the program to evaluate to zero. Equivalently,* $\mathbf{p} \in \mathbf{R}^{nd}$ *is in* general position *if there is no test determinant* f *such that* $f(\mathbf{p}) = 0$ *for* f *not identically zero.*

2.3 Problem definition

Given is an algorithm that is correct for input in general position, in other words, it produces the exact solution for non-degenerate inputs. We wish to systematically perturb an arbitrary input $\mathbf{p}$ into some different instance $\mathbf{p}(\epsilon)$, where ϵ is an infinitesimal symbolic variable, on which the original algorithm can run and still produce some useful output.

A valid perturbation guarantees the following properties. First, that the output on $\mathbf{p}(\epsilon)$ is the same as that on $\mathbf{p}$, if the latter is in general position. Otherwise, that it is arbitrarily close to the correct solution, provided that the mapping from input to solution space is continuous, where both spaces are equipped with some appropriate metric. In this case, the solution is restored from the actual output by setting ϵ to zero or taking its limit as it goes to zero. In practice, this involves some kind of post-processing, as in constructing a convex hull with no interior points in its faces. Then a post-processing phase must remove the artificially created faces and the corresponding points, in other words,

all faces forming infinitesimal volumes with the faces of the exact solution. Post-processing in a different context is described in [Do].

Lastly, for degenerate $\mathbf{p}$ and non-continuous mappings, there is no requirement on the output obtained on $\mathbf{p}(\epsilon)$. This case is of little practical interest in geometry because it is usually possible to define some topology under which continuity is achieved. For the Convex Hull problem, for instance, that looks inherently discrete, we could take the distance of two hulls in the output space to be proportional to their symmetric difference.

In this paper we content ourselves with a characterization shown in [EmCa] to be equivalent to the above requirements.

Definition 2.2 *We say that a perturbation is* valid *if, given input instance* $\mathbf{p}$*, it defines a new instance* $\mathbf{p}(\epsilon)$ *which is (i) in general position, (ii) arbitrarily close to* $\mathbf{p}$ *in the input space topology and such that, (iii) when* $\mathbf{p}$ *is not degenerate, the computation paths of the algorithm on the two inputs are identical.*

Each perturbation defines a pre-processing stage that alters the input parameters, but the new parameters affect only the computation of the test determinants. For the implementor each predicate may be regarded as a black box which, given a set of parameter indices and the respective values, returns either -1 or 1 (or 0 before degeneracies are removed); these black boxes are the only parts that must be rewritten.

2.4 Infinitesimals

The perturbations make use of a positive infinitesimal variable ϵ. There are two equivalent ways to think of it. Formally, it is an extension to the reals, larger than zero but smaller than any positive real number. For our purposes, it suffices to regard it as a positive real variable that is sufficiently small so that it avoids all roots of a given set of polynomials. Regardless of the preferred point of view, the sign of a polynomial in ϵ is the sign of the coefficient of least degree.

One issue is whether the original algorithm can run on perturbed input, and, in particular, whether it is going to halt. The above argument asserts that the computation path on $\mathbf{p}(\epsilon)$ is the same as if ϵ was substituted by a sufficiently small positive real value.

3 Previous work

The most naive approach is to handle each special case separately, which is tedious for implementors and unattractive for theoreticians. Random perturbations have been popular, yet they provide no certainty. Dantzig's [Da] symmetry breaking rules in linear programming is regarded as the precursor of current systematic approaches.

Edelsbrunner and Mücke generalize in [EdMu] a technique called Simulation of Simplicity (SoS for short), already presented in [EdGu], [Ed] and [EdWa]. Every input parameter $p_{i,j}$ is perturbed into

$$p_{i,j}(\epsilon) = p_{i,j} + \epsilon^{2^{i\delta - j}},$$

where $\delta > d$ and d is the dimension of the geometric space where the input objects lie. The perturbation is infinitesimal due to symbolic variable ϵ which, although never evaluated, is assumed positive and arbitrarily small; it is also conceptual in the sense that the computation remains numeric as ϵ is never explicitly used. SoS applies to a wide range of geometric primitives, including those examined in this paper, except for an inconsistency in the case of the InSphere predicate discussed in 5.4. Its main drawback is that deciding the sign of a $d \times d$ perturbed determinant requires $\Omega(2^d)$ steps in the worst case.

Yap in [Ya87] deals with the more general setting in which branching occurs at arbitrary rational expressions and proposes a method which is equivalent to an infinitesimal perturbation, as proven in [Ya88]. For input $\mathbf{p} = (p_1, \ldots, p_n)$, each test polynomial $f(\mathbf{p})$ is associated with the infinite list

$$S(f) \;=\; (f, f_{w_1}, f_{w_2}, \ldots)$$

where f_{w_k} is the partial derivative of f with respect to the k-th power product in the input variables. The sign of f is taken to be the sign of the first polynomial in $S(f)$ with a non-zero value, which can always be found after a finite number of evaluations. In the worst case that all partial derivatives have to be computed, the complexity per test is $\Omega(d^n)$.

Dobrindt proposed an efficient scheme to cope with degenerate intersections between a convex and a general polyhedron, in [Do]. The convex polyhedron is perturbed symbolically both "outwards" and "inwards" so that enough information is available for fully specifying the intersection in a post-processing phase.

A *structural* perturbation for a motion-planning algorithm, in which the input objects are the semi-algebraic sets describing the obstacles, is given by Canny in [Ca]. He uses towers of infinitesimals to eliminate degeneracies while preserving essential properties of the sets, namely emptiness and number of connected components.

Emiris and Canny presented in [EmCa] a direct infinitesimal perturbation that constitutes the first efficient scheme. For geometric algorithms, every coordinate $p_{i,j}$ is deterministically perturbed into

$$p_{i,j}(\epsilon) = p_{i,j} + \epsilon i^j, \qquad (1)$$

where ϵ is a symbolic infinitesimal. The perturbation applies to the Orientation and Transversality primitives with complexity overheads $\mathcal{O}(1)$ and $\mathcal{O}(d^{1+\alpha})$ under the algebraic and bit models respectively, for an arbitrarily small positive constant α, and requires no symbolic computation. In the general case that branching occurs at arbitrary rational expressions, randomization is traded for efficiency and the i-th perturbed parameter is

$$p_i(\epsilon) = p_i + \epsilon r_i,$$

where r_i is a random integer uniformly chosen from a sufficiently large interval. Let D denote the maximum degree in the input parameters of any polynomial in the program and $\phi(n)$ the program's bit complexity. The size of the interval from which the random quantities are chosen increases with D, $\phi(n)$ and the desired success rate. In this context, the perturbation is successful when it removes all degeneracies. The algebraic complexity overhead is $\mathcal{O}(D^{1+\alpha})$ and the bit complexity overhead is $\mathcal{O}(\phi^{2+\alpha}(n))$, where α is as above. Hence, the overall running-time of an algorithm on perturbed input is bounded by a polynomial in the original complexity.

4 The perturbation

The following section extends the application of deterministic perturbation (1) to Ordering and InSphere, thus covering all four predicates in the model. Moreover, a more efficient perturbation is defined and shown valid with respect to the Orientation and Transversality predicates.

The input consists of n objects, most conveniently thought of as points, each described by d coordinates, for $d < n$, where at least a constant fraction of these objects are distinct. Clearly, the bit complexity of the perturbation quantities affects the scheme's efficiency. Seidel [Se] suggested a variant of (1) that is optimal in this sense:

$$p_{i,j}(\epsilon) = p_{i,j} + \epsilon(i^j \bmod q), \tag{2}$$

where $1 \leq i \leq n$, $1 \leq j \leq d$, ϵ denotes the symbolic infinitesimal and q is the smallest prime that exceeds n. Then the bit size of the perturbation quantities is bounded by $\log n$, which is the best we can hope for, since there must be at least n distinct such quantities. Perturbation (2), like its predecessor, defines n vectors, every d of which are linearly independent. This is the crucial property when proving validity with respect to predicates Orientation and Transversality below.

We mention another two perturbations. As shown later, the validity of (1) and (2) is related to the non-singularity of Vandermonde matrices. Alternatively, we may rely on the non-singularity of a Cauchy matrix:

$$p_{i,j}(\epsilon) = p_{i,j} + \epsilon \frac{1}{i+j-1} \tag{3}$$

defines a valid scheme with respect to Orientation and Transversality with no improvement, however, upon the efficiency of (2). Another option is to use the first n primes, denoted as $q_1, \ldots, q_n$:

$$p_{i,j}(\epsilon) = p_{i,j} + \epsilon(q_i^j). \tag{4}$$

The bit complexity of the perturbation quantities is then $\mathcal{O}(d \log n)$. This paper uses (4) only to cover some limitation of schemes (1) and (2), as discussed in 5.4. One should keep in mind that consistency requires that exactly one scheme is applied to all predicates of a specific algorithm.

5 The predicates

5.1 Ordering

This predicate decides the order of two parameters expressing the k-th coordinate of the i_1-th and i_2-th input objects. It is the most basic primitive test and is presented here in order to extend the applicability of our perturbation schemes in a consistent manner. Breaking ties along the k-th coordinate ensures, for instance, that no two points lie on a common hyperplane perpendicular to the k-th axis. On perturbed input the predicate decides the sign of

$$p_{i_1,k}(\epsilon) - p_{i_2,k}(\epsilon) = p_{i_1,k} + \epsilon i_1^k - p_{i_2,k} - \epsilon i_2^k.$$

The most significant term does not involve ϵ. If this term is zero, though, the factors of the infinitesimal must be compared, which comes down to comparing i_1 against i_2. Since all indices are distinct the predicate always returns a non-zero answer. Notice that this is the lexicographic ordering.

The implementation requires in the worst case, an extra constant-time check, thus doubling the running time of the predicate. Similarly, under the bit model, the extra comparison adds a $\mathcal{O}(\log n)$ factor, which is upper bounded by the original bit complexity. Perturbation (2) is also valid and as efficient for $k = 1$. We have demonstrated the following

Proposition 5.1 *Perturbation (1) is valid with respect to the Ordering predicate and does not change the asymptotic running-time complexity of this predicate in the algebraic as well as the bit model. The same holds for perturbation (2) along the first coordinate.*

Unfortunately, if we compare along some general k-th coordinate, even validity may not hold for the second scheme.

5.2 Orientation and Transversality

We apply perturbation (2) to both of these predicates together. Orientation decides, given a query point $p_{i_{d+1}}$ and a hyperplane in $\mathbf{R}^d$ spanned by points $p_{i_1}, \ldots p_{i_d}$, in which halfspace the point lies. A degeneracy occurs exactly when $p_{i_{d+1}}$ lies on the hyperplane. The predicate is formulated as a test of a determinant sign; the relevant matrix is like Λ_{d+1} below with the column of ones as the rightmost column.

$$\Lambda_{d+1} = \begin{bmatrix} 1 & p_{i_1,1} & p_{i_1,2} & \cdots & p_{i_1,d} \\ 1 & p_{i_2,1} & p_{i_2,2} & \cdots & p_{i_2,d} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & p_{i_{d+1},1} & p_{i_{d+1},2} & \cdots & p_{i_{d+1},d} \end{bmatrix}.$$

The perturbed matrix $\Lambda_{d+1}(\epsilon)$ contains the corresponding perturbed parameters and its determinant is a polynomial in ϵ:

$$\begin{aligned} det\Lambda_{d+1}(\epsilon) \;&=\; det\Lambda_{d+1} + \\ &+ (\epsilon^k \; terms, 1 \le k \le d-1) \;+\; \epsilon^d \, det\overline{V_{d+1}}. \end{aligned}$$

The non-singularity of Λ_{d+1} rests on the non-singularity of $\overline{V_{d+1}}$, which is a $(d+1) \times (d+1)$ matrix whose entries are the moduli with q of the corresponding Vandermonde entries, where $q > n$ is the prime in the definition of scheme (2). To show that $det\overline{V_{d+1}}$ does not vanish, it suffices to prove that it is not congruent to 0 mod q.

$$\begin{aligned} det\overline{V_{d+1}} \;&\equiv\; \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^d \\ 1 & i_2 & i_2^2 & \cdots & i_2^d \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & i_{d+1} & i_{d+1}^2 & \cdots & i_{d+1}^d \end{vmatrix} \\ &\equiv \prod_{\substack{k=2,\, l=1 \\ k>l}}^{d} (i_k - i_l) \not\equiv 0 \pmod q. \end{aligned}$$

Transversality determines the orientation of d points in $\mathbf{R}^{d-1}$, given by their homogeneous coordinates, and is expressed as the sign of the determinant of

$$\Delta_d = \begin{bmatrix} p_{i_1,1} & p_{i_1,2} & \cdots & p_{i_1,d} \\ p_{i_2,1} & p_{i_2,2} & \cdots & p_{i_2,d} \\ \vdots & \vdots & & \vdots \\ p_{i_d,1} & p_{i_d,2} & \cdots & p_{i_d,d} \end{bmatrix}.$$

This matrix also comes up in a dual context, as in [EdGu] where the input objects are lines on the plane, each defined by three parameters, and Transversality decides on which side of the third line the intersection of the first two lies. In every instance, a degeneracy occurs exactly when Δ_d is singular. Calculations derive

$$\begin{aligned} &det\Delta_d(\epsilon) \;= \\ &\quad = \; det\Delta_d \;+\; (\epsilon^k \; terms, 1 \le k \le d-1) \;+\; \epsilon^d \; det\overline{U_d} \end{aligned}$$

where the entries of $\overline{U_d}$ are the moduli of an analogous Vandermonde matrix such that

$$det\overline{U_d} \equiv \prod_{k=1}^{d} i_k \prod_{\substack{k=2,\, l=1 \\ k>l}}^{d} (i_k - i_l) \not\equiv 0 \pmod q.$$

Lemma 5.2 *There exists a positive real constant ϵ_0 such that, for every positive real $\epsilon < \epsilon_0$, every $\Lambda_{d+1}(\epsilon)$ and $\Delta_d(\epsilon)$ matrix occurring at a branching node of a given algorithm after perturbation (2) is non-singular and its determinant has constant sign.*

Proof From the theory of polynomials over ordered fields, the ϵ-polynomials, since they are not identically zero, have a set of roots of codimension 1, i.e. of dimension 0. Any algebraic set of roots whose dimension is at most 1, is either the entire real line or the union of a finite number of points. Therefore $det\Lambda_{d+1}(\epsilon)$ and $det\Delta_d(\epsilon)$ each has a finite number of roots. It suffices to let ϵ_0 be the minimum positive such root over all test determinants in the program. □

The sign of $det\Lambda_{d+1}(\epsilon)$ and $det\Delta_{d+1}(\epsilon)$ is the sign of the least significant term in the respective polynomial. One way to compute it, adopted by SoS, is to calculate directly all terms, starting with the one of least degree, until finding one that does not vanish. Fortunately, our technique lends itself to a more efficient trick that reduces the computation of the perturbed determinant to a characteristic polynomial computation.

$$\begin{aligned} &det\Lambda_{d+1}(\epsilon) \\ &= \frac{1}{\epsilon} det \left(\begin{bmatrix} 0 & p_{i_1,1} & \cdots & p_{i_1,d} \\ \vdots & \vdots & & \vdots \\ 0 & p_{i_{d+1},1} & \cdots & p_{i_{d+1},d} \end{bmatrix} + \epsilon \, \overline{V_{d+1}} \right) \\ &= \frac{1}{\epsilon} \; det \left(L \;+\; \epsilon \overline{V_{d+1}} \right) \\ &= \frac{1}{\epsilon} det(-\overline{V_{d+1}}) \; det((-\overline{V_{d+1}}^{-1})L - \epsilon \, I_{d+1}) \\ &= \frac{1}{\epsilon} (-1)^{d+1} det\overline{V_{d+1}} \; det(M - \epsilon \, I_{d+1}). \end{aligned}$$

Similarly,

$$det\Delta_d(\epsilon) \;=\; det(-\overline{U_d}) \; det(N - \epsilon \, I_d),$$

where L and M are implicitly defined, I_k stands for the unit matrix of k-th order and $N = (-\overline{U_d}^{-1})\,\Delta_d$. Let $MM(k)$ denote the number of algebraic operations required to multiply two $k \times k$ matrices, which is strictly larger than k^2.

Lemma 5.3 *Computing the sign of the determinants of perturbed matrices $\Lambda_{d+1}(\epsilon)$ and $\Delta_d(\epsilon)$ obtained from perturbation (2) has complexity $\mathcal{O}(MM(d))$ under the algebraic model.*

Proof Both predicates perform certain matrix manipulation operations, each taking $\mathcal{O}(MM(d))$ steps, including the characteristic polynomial computation, due to an algorithm by Keller-Gehrig [KG]. □

Proposition 5.4 *Perturbation (2) is valid and does not change the running-time complexity of the Orientation and Transversality predicates under the algebraic or the bit model. These results hold regardless of whether long or modular arithmetic is used.*

Proof At branching nodes, the sign of the perturbed determinant is taken to be the sign of the lowest order non-vanishing term in ϵ, which is the original determinant if this is non-zero. Otherwise, we simulate an artificial non-degenerate situation in which the input points are arbitrarily close to the original ones. Lemma 5.2 proves the consistency of our answers as ϵ approaches to zero on the positive real axis.

Lemma 5.3 asserts that the implementation takes $\mathcal{O}(MM(d))$ operations which is the original complexity of the algorithm in the algebraic model since it had to compute a determinant of order d.

With regard to bit complexity, the original running time is $\Omega(MM(d)M(d\log n))$ because at least a constant fraction of operations involves $\Theta(d\log n)$-bit numbers. After perturbing, the characteristic polynomial has coefficients of size $\mathcal{O}(d\log n)$, hence the bit complexity remains asymptotically unchanged.

If instead of long arithmetic, modular arithmetic is used in conjunction with the Chinese Remainder theorem, the asymptotic complexity is again unaffected. The number of finite fields required depends on the size of the final answer, which is asymptotically the same. Mapping the various quantities into finite fields and computing in them is as costly before and after the perturbation. The third phase that computes the final answer from its moduli may be slower by a factor of $\mathcal{O}(d)$ when computing the coefficients of the characteristic polynomial, but this is not the dominant factor. □

5.3 InSphere

We apply (1) to this predicate which decides, given $d+2$ points, whether the $(d+2)$-nd point lies in the interior of the higher-dimensional sphere defined by the first $d+1$ points in $\mathbf{R}^d$. It can be reduced to testing the sign of a determinant. First, lift all points to the surface of a paraboloid in $\mathbf{R}^{d+1}$ by adding a $(d+1)$-st coordinate equal to the sum of the squares of the d coordinates defining each point. The original space is a d-dimensional hyperplane which the paraboloid touches at the origin.

Let $p_{i_1}, p_{i_2}, \ldots, p_{i_{d+1}}$ be the points defining the sphere, their liftings also define a hyperplane H in $\mathbf{R}^{d+1}$. The query point $p_{i_{d+2}}$ lies within the sphere if and only if its lifted image lies below H, in other words to the same side of H as the original points. A degeneracy occurs exactly when $p_{i_{d+2}}$ lies on the sphere or, equivalently, on H. We concentrate on $,_{d+2}$ which is obtained from the proper matrix by conveniently moving the column of ones to the first column.

$$,_{d+2} = \begin{bmatrix} 1 & p_{i_1,1} & p_{i_1,2} & \cdots & p_{i_1,d} & \sum_{j=1}^{d} p_{i_1,j}^2 \\ 1 & p_{i_2,1} & p_{i_2,2} & \cdots & p_{i_2,d} & \sum_{j=1}^{d} p_{i_2,j}^2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & p_{i_{d+1},1} & p_{i_{d+1},2} & \cdots & p_{i_{d+1},d} & \sum_{j=1}^{d} p_{i_{d+1},j}^2 \\ 1 & p_{i_{d+2},1} & p_{i_{d+2},2} & \cdots & p_{i_{d+2},d} & \sum_{j=1}^{d} p_{i_{d+2},j}^2 \end{bmatrix}.$$

Eliminating degeneracies for the particular matrix could be achieved by the "cheap trick" of [EdMu] which perturbs the points on the higher-dimensional paraboloid, by perturbing the sum of squares as if it was an additional coordinate. However, this may lead to inconsistencies, if the algorithm uses another predicate such as Λ_{d+1} in conjunction with $,_{d+2}$. Consider, for instance, deciding the relative position of a line and a circle which touch at two coincident points p_1 and p_2, by using the Orientation predicate on the line and p_1 and the InSphere predicate on the circle and p_2. Hence we need to prove that the proper perturbation works for $,_{d+2}$.

The determinant of the perturbed matrix $,_{d+2}(\epsilon)$ is

$$det,_{d+2}(\epsilon) = det,_{d+2} + \\ + (\epsilon^k \; terms, 1 \le k \le d+1) + \epsilon^{d+2} det W_{d+2},$$

where W_{d+2} resembles a Vandermonde matrix:

$$W_{d+2} = \begin{bmatrix} 1 & i_1 & \cdots & i_1^d & \sum_{j=1}^{d} i_1^{2j} \\ 1 & i_2 & \cdots & i_2^d & \sum_{j=1}^{d} i_2^{2j} \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & i_{d+2} & \cdots & i_{d+2}^d & \sum_{j=1}^{d} i_{d+2}^{2j} \end{bmatrix}.$$

To prove that $det,_{d+2}(\epsilon)$ is not identically zero, it suffices to show that W_{d+2} is not singular. Assume that it is. Then there exists a non-zero $(d+2)$-vector $\mathbf{z}$ in its kernel. Fix $\mathbf{z} = [z_0 \; z_1 \; \ldots \; z_{d+1}]$. Then the following univariate polynomial in x,

$$z_0 + x z_1 + x^2 z_2 + \cdots + x^d z_d + \sum_{j=1}^{d} x^{2j} z_{d+1},$$

has at least $d+2$ distinct positive integer solutions, namely $i_1, \ldots, i_{d+2}$.

To derive a contradiction consider Descartes' rule of sign, as presented by Collins and Loos in [CoLo]. It states that the number of sign variations of a univariate

polynomial's coefficients exceeds the number of positive zeros, multiplicities counted, by an even non-negative integer. To define the number of sign variations of a finite sequence of real numbers $< u_1, \ldots, u_s >$, let $< u'_1, \ldots, u'_t >$ be the subsequence of all non-zero elements. Then the number of variations is the number of consecutive element pairs u_k, u_{k+1}, for $1 \leq k < t$, such that the product $u_k u_{k+1}$ is negative.

Applying Descartes' rule we conclude that the number of sign variations must be at least equal to $d+2$, hence there must exist at least $d+3$ non-zero distinct coefficients. We rewrite the x-polynomial in its standard form:

$$z_0 + x z_1 + x^2(z_2 + z_{d+1}) + x^3 z_3 + \\ + x^4(z_4 + z_{d+1}) + \cdots + \sum_{j=\lceil d/2 \rceil}^{d} x^{2j} z_{d+1}.$$

Obviously, there are at most $d+2$ distinct coefficients, which leads to a contradiction. Therefore the hypothesis about the existence of vector $\mathbf{z}$ is false, which implies that the kernel of the linear transformation contains only zero. Equivalently, the matrix of the transformation, W_{d+2}, is non-singular, and hence $det,\ _{d+2}(\epsilon)$ not identically zero.

Lemma 5.5 *There exists a positive real constant ϵ_0 such that, for every positive real $\epsilon < \epsilon_0$, the determinant of $,\ _{d+2}(\epsilon)$, obtained from perturbation (1), is non-zero and has constant sign.*

Proof Similar to that of lemma 5.2. □

This argument cannot be repeated with the second perturbation, hence the validity proof fails for (2).

We still have to tackle the question of implementing the InSphere predicate, under the first perturbation. Taking advantage of the linearity of the determinant in the last column, we reduce the computation to two parts, each being reduced to a characteristic polynomial computation. Omitting the details, here are the final expressions.

$$\begin{aligned} det,\ _{d+2}(\epsilon) \ &= \\ &= \ det(-W_{d+2})\ det((-W_{d+2}^{-1})G_1 - \epsilon I_{d+2}) \\ &+ \ \frac{1}{\epsilon}\, det(-V_{d+2})\ det((-V_{d+2}^{-1})G_2 - \epsilon I_{d+2}), \end{aligned}$$

where W_{d+2} was given above and shown non-singular, V_{d+2} is a $(d+2) \times (d+2)$ Vandermonde matrix like the one used with Orientation, with

$$G_1 \ = \ \begin{bmatrix} 0 & p_{i_1,1} & \cdots & p_{i_1,d} & 2\sum_{j=1}^{d} p_{i_1,j}\, i_1^j - i_1^{d+2} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & p_{i_{d+2},1} & \cdots & p_{i_{d+2},d} & 2\sum_{j=1}^{d} p_{i_{d+2},j}\, i_{d+2}^j - i_{d+2}^{d+2} \end{bmatrix}$$

and

$$G_2 = \begin{bmatrix} 0 & p_{i_1,1} & \cdots & p_{i_1,d} & \sum_{j=1}^{d} p_{i_1,j}^2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & p_{i_{d+2},1} & \cdots & p_{i_{d+2},d} & \sum_{j=1}^{d} p_{i_{d+2},j}^2 \end{bmatrix}.$$

Lemma 5.6 *Computing the sign of the determinant of the perturbed matrix $,\ _{d+2}(\epsilon)$ obtained from perturbation (1), under the algebraic model, takes $\mathcal{O}(MM(d))$ time.*

Proof Similar to that of lemma 5.3. □

Proposition 5.7 *Perturbation (1) is valid and does not affect the complexity of the InSphere predicate under the algebraic model. Under the bit model, the complexity increases by a factor $\mathcal{O}(d^{1+\alpha})$, for an arbitrarily small positive constant α. Both results hold regardless of whether long or modular arithmetic is used.*

Proof Similar to that of proposition 5.4. Intuitively, the size of the answer in the original algorithm is $\mathcal{O}(d \log n)$, while after perturbing, it is $\mathcal{O}(d^2 \log n)$. Again, we lower bound the input parameters by $\log n$; bit operations cost $\mathcal{O}(M(d \log n))$, while with perturbed quantities they cost $\mathcal{O}(M(d^2 \log n))$, which shows intuitively the source of the complexity overhead. The argument from proposition 5.4 can be repeated in the case of modular arithmetic. □

Notice that the two characteristic polynomial computations involve similar matrices and hence the respective implementations may be combined to yield a constant factor speedup, especially if some simpler algorithm is chosen over Keller-Gehrig's optimal one.

5.4 Other predicates

It is a simple matter to extend the model to capture more primitive tests and hence a wider class of algorithms, by establishing the validity of our schemes with respect to the new predicates. Moreover, the input vectors may describe other kinds of objects than points, such as hyperplanes, thus opening several algorithms to our approach, as mentioned in [EdMu]. That paper also lists several predicates for each one of which one of our schemes is valid.

Predicates that decide on the relative position of *derived* objects may pose a limitation to our method. Consider, for instance, the two-dimensional ham-sandwich algorithm in [EdWa] with lines on the plane being the input objects and their intersection points being the derived objects. Here are the three predicates called by the algorithm, assuming that the lines passed to a test or those defining the points in a test are all distinct. Deciding whether a point lies above or below a line; comparing the x-coordinates of two points; and comparing the distances of two points from a line.

Applying scheme (1) to the points removes all degeneracies but it is not clear that this does not create some geometrically inconsistent configuration. Applied to the input lines, SoS successfully perturbs them into general position; however, perturbation (1) fails for the second test, while perturbation (4) is valid for the second but not for the third test.

6 Conclusion

We have defined two perturbation schemes that eliminate degeneracies in the context of certain geometric predicates and proved their validity. The merit of these schemes is twofold. First, their simplicity makes them attractive for practical use. Given a program that works for input in general position, the only parts that must be changed when introducing our perturbations are those that compute the value of some predicate. Moreover, although defined in terms of an infinitesimal symbolic variable, neither scheme requires any symbolic computation. The first author has actually implemented the bit-optimal scheme (2) for a Convex Hull algorithm in general dimension.

The second advantage of our methods over previous ones is their efficiency. Under the algebraic model, the complexity overhead due to either of our schemes is a small constant; the same holds under the bit model for Orientation and Transversality and the bit-optimal scheme. For the InSphere predicate we offer perturbation (1) which incurs an extra bit complexity factor $\mathcal{O}(d^{1+\alpha})$ for an arbitrarily small positive constant α that accounts for the logarithmic factors.

The predicates studied cover most important algorithms in computational geometry. It is interesting to look for another scheme that optimizes the bit complexity overhead for all predicates, including InSphere. We may, lastly, attempt the same strategy in attacking the problem of degeneracy in other areas such as geometric modeling.

Acknowledgment

We wish to thank Raimund Seidel for pointing out the InSphere predicate, as well as for several useful discussions.

References

[AHU] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[Ca] Canny J., Computing Roadmaps of Semi-Algebraic Sets, *Proc. 9th AAECC*, New Orleans, 1991.

[CoLo] Collins G.E. and R. Loos, Real Zeros of Polynomials, *Computer Algebra: Symbolic and Algebraic Computation*, ed. B. Buchberger et.al. Springer-Verlag, Wien, pp. 83-94, 1982.

[Da] Dantzig G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton, 1963.

[Do] Dobrindt K., Algorithmen für Polyeder, Master's thesis (in German), Fachbereich Informatik, Universität Saarbrücken, Germany, May 1990.

[Ed] Edelsbrunner H., *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[EdGu] Edelsbrunner H. and L.J. Guibas, Topologically Sweeping an Arrangement, *Proc. 18th Annual ACM STOC*, Berkeley, pp. 389-403, 1986.

[EdMu] Edelsbrunner H. and E.P. Mücke, Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms, *ACM Trans. Graphics*, Vol. 9, No. 1, pp. 67-104, 1990.

[EdWa] Edelsbrunner H. and R. Waupotitsch, Computing a ham-sandwich cut in two dimensions, *J. Symbolic Comput. 2*, pp. 171-178, 1986.

[Em] Emiris I., An Efficient Approach to Removing Geometric Degeneracies, Master's thesis, Computer Science Division, UC Berkeley, May 1991.

[EmCa] Emiris I. and J. Canny, A General Approach to Removing Degeneracies, *Proc. 32nd Annual IEEE FOCS*, San Juan, pp. 405-413, 1991.

[KG] Keller-Gehrig W., Fast Algorithms for the Characteristic Polynomial, *Theor. Comp. Sci.*, Vol. 36, pp. 309-317, 1985.

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

[Se] Seidel R., private communication, 1991.

[Ya87] Yap C.-K., Symbolic treatment of geometric degeneracies, *Proc. 13th IFIP Conf. on Sys. Modeling and Optimization*, Tokyo, pp. 348-358, 1987.

[Ya88] Yap C.-K., A geometric consistency theorem for a symbolic perturbation scheme, *Proc. 4th ACM Symp. on Comp. Geometry*, Urbana, pp. 134-142, 1988.