Discrete Comput Geom 19:1–17 (1998)

Discrete & Computational
Geometry

The Nature and Meaning of Perturbations in Geometric Computing*

R. Seidel†

Computer Science Division, 673 Soda Hall, University of California Berkeley,
Berkeley, CA 94720-1776, USA
seidel@cs.berkeley.edu
and
Fachbereich 14, Informatik, Universität des Saarlandes,
Postfach 151150, D-66041 Saarbrücken, Germany

Abstract. This paper addresses some fundamental questions concerning perturbations as they are used in computational geometry. How does one define them? What does it mean to compute with them? How can one compute with them? Is it sensible to use them?

We define perturbations to be curves, point out that computing with them amounts to computing with limits, and (re)derive some methods of computing with such limits automatically. In principle, a line can always be used as a perturbation curve. We discuss a generic method for choosing such a line that is applicable in many situations.

1. Introduction

When faced with the problem of geometric degeneracy, a typical computational geometry paper, talk, or lecture will simply appeal to "standard perturbation methods." This may be followed by a reference, such as [7], [20], or [9], but usually not much more discussion is offered. This may be okay, were there not the curious fact that when one asks a typical computational geometer for a precise definition of "perturbation," one will usually not receive a satisfactory answer ("*You perturb the input a little bit.*" "*You move the input points by an infinitesimal amount.*" etc.). Consulting the main references does not guarantee satisfaction either. Usually one finds a lengthy discussion on "degeneracies" and the distinction between "problem-induced" and "algorithm-induced" ones, and then

* A preliminary version of this paper has appeared in *Proc. 11th STACS*, Lecture Notes in Computer Science, vol. 775, Springer-Verlag, Berlin, 1994.

† Supported by the NSF Presidential Young Investigator Award CCR-9058440.

perturbations are defined, either by what properties they are supposed to have (and not by what they are supposed to be), or as sequences satisfying some technical, rather unintuitive conditions.

Why this peculiar situation? The answer, I believe, lies in the somewhat dubious motivation for using perturbations: Here I have a program that solves some problems in all nondegenerate instances (whatever nondegenerate means). Give me some automatic, strictly syntactic method that transforms my program into one that solves the problem in all instances.

A reasonably cautious researcher will almost immediately raise his (or her) eyebrows. How can we transform, by purely syntactic means, a program that does not always work into one that always works? This seems like wishful thinking and is very unlikely to succeed in general. But, as it turns out, the method of perturbations comes surprisingly close to achieving just that. The transformed program always "works." However, it does not always solve the original problem, but it solves some related problem. The exact nature of this related problem, though, and its relationship to the original problem, is usually left unclear.[1] However, here exactly seems to be the crux of the difficulty of understanding, defining, and using perturbations.

It is the goal of this paper to elucidate some of these issues. In Section 2 we define perturbations (for us they are just curves). Before ever mentioning algorithms or programs, we show how a perturbation scheme can be used to transform a *problem mapping* into a *perturbed problem mapping* via a simple limit construction. We consider under what circumstances the original mapping F and and the perturbed mapping $\overline{F}$ agree, and hence computing $\overline{F}$ gives the same result as computing F. Finally, we discuss how for certain models of computation any program that computes F "almost everywhere" can be purely syntactically transformed into a program that computes $\overline{F}$ everywhere (which is great if F and $\overline{F}$ agree).

Section 3 considers the choice of perturbation curves. It comes as no big suprise that the simplest of all possible nontrivial curves, namely a straight line, can always be chosen to do the job. A somewhat surprising finding though shows that, considering almost any geometric problem, essentially the same line can be used for all possible inputs. The finding can be paraphrased as "*if you know one nondegenerate input, then you have a good perturbation for all inputs*." We discuss the ramifications of this insight, in particular, the problem of producing nondegenerate inputs.

In Section 4 we discuss previous perturbation schemes and how they fit into our framework. This is almost immediate for the SoS scheme of Edelsbrunner and Mücke [7] and for the efficient linear scheme of Emiris and Canny [9], [8]. It is slightly more involved for the symbolic scheme of Yap [20], [21], however, it leads to a consistency proof that is maybe simpler than the ones offered in the original papers.

The last section deals with the shortcomings of the perturbation method, and to some extent questions the wisdom of using it.

This paper does not contain many new results. It is rather meant mainly as an elucidation and clarification of previous ideas. The topic of perturbations is one of the very few ones that, in my experience, can cause heated, opinionated discussions among compu-

[1] For instance, it is *a priori* not clear whether the transforms of two different programs for the same original problem solve the same related problem.

tational geometers. Thus I apologize in advance if some of my statements have turned out more pointed than they should be.

2. The Framework

2.1. *Perturbations*

In this section we consider the computation of functions F from some *input space* $\mathcal{I}$ to some *output space* $\mathcal{O}$. We will assume that $\mathcal{I}$, as well as $\mathcal{O}$, is endowed with some topology, which allows for the notion of a limit. Typically, $\mathcal{I}$ will be $\mathbb{R}^N$ with the usual Euclidean topology; $\mathcal{O}$ will be $\mathbb{R}^M$ with Euclidean topology, or some finite set with the discrete topology, or the product or direct sum of such spaces.

As typical examples consider *CHS*, the *convex hull sequence* function, which given the coordinates of a sequence S of n points $(x_1, y_1), \ldots, (x_n, y_n)$ in the plane asks for the sequence of indices of the points that constitute the vertices of the convex hull of S, and consider *CHA*, the *convex hull area* function, which, given the sequence S just asks for a single real number, namely the area of the convex hull of S. For both functions the input space $\mathcal{I}$ is $\mathbb{R}^{2n}$ with its usual topology, where the input sequence S is encoded as the $(2n)$-tuple $q = (x_1, y_1, \ldots, x_n, y_n)$. For *CHA* the output space $\mathcal{O}$ is just $\mathbb{R}$ with the usual topology. For *CHS* the output space $\mathcal{O}$ is the (finite) set of all sequences of distinct integers from $\{1, \ldots, n\}$ of length at most n with the discrete topology.

Definition 1. For a point $q \in \mathcal{I}$ we define a *perturbation of* q to be any curve $q(\,)$ beginning in q, in other words, the image of a continuous mapping $q(\,)\colon [0, \infty) \mapsto \mathcal{I}$ with $q(0) = q$. A *perturbation scheme* Q assigns to every point $q \in \mathcal{I}$ a perturbation $q(\,)$. [2]

Definition 2. A perturbation scheme Q induces for every function $F\colon \mathcal{I} \mapsto \mathcal{O}$ a *perturbed function* $\overline{F}^Q\colon \mathcal{I} \mapsto \mathcal{O}$, defined by

$$\overline{F}^Q(q) = \lim_{\varepsilon \to 0^+} F(q(\varepsilon)).$$

We will assume that this limit exists. (If it does not, there is little sense in applying the method of perturbations.) Since the scheme Q is often clear from context, we will sometimes omit the superscript Q and just write $\overline{F}$ for the perturbed function. Keep in mind, though, that the perturbed function does depend on the scheme Q. In general, for two different perturbation schemes Q and Q' the perturbed functions $\overline{F}^Q$ and $\overline{F}^{Q'}$ will be different.

When do F and its perturbed function $\overline{F}$ agree? Here is a simple and obvious, but useful condition.

[2] For the sake of better readability we forego a more precise notation where a perturbation scheme π assigns to each point q a curve π_q.

Lemma 3. *If F is continuous at q, then $F(q) = \overline{F}(q)$.*

Note that the statement in the lemma holds for any perturbation scheme. Also note, that if F is not continuous at q, very little, if anything at all, can be said in general about the relationship between $F(q)$ and $\overline{F}(q)$. Of course our hope is that there is some reasonable relationship, because the whole idea of using the "perturbation method" is to compute $\overline{F}(q)$ instead of $F(q)$.

Why is this useful? Frequently, instead of writing a program that computes F for all inputs, it is much easier to write a program Π that correctly computes F for *almost all* inputs (ignore the special "degenerate" cases!). This partially working program Π can then rather straightforwardly be transformed into a program $\overline{\Pi}$ that computes $\overline{F}$ for all inputs. In other words, in order to make Π work for all inputs one simply redefines the problem that it is supposed to solve.

This certainly seems like a dubious way of proceeding, but is often less crazy than it looks. If, for instance, F is continuous everywhere, then $\overline{F} = F$ everywhere and the program $\overline{\Pi}$ computes the correct thing. This happens, for instance, in the convex hull area function *CHA* mentioned above. But even if F is not continuous for some input q, computing $\overline{F}(q)$ can yield enough information to recover $F(q)$ relatively easily. For instance, the convex hull sequence function *CHS* mentioned above is discontinuous for inputs q representing planar point sets that have more than two points collinear on a convex hull edge. However, it is easy to see that in this case $CHS(q)$ must be a subsequence of $\overline{CHS}(q)$; the extra elements correspond to vertices in the middle of edges and can be discovered and removed easily in a postprocessing step.

Not all discontinuous functions admit such an easy postprocessing step. We will dwell no further on this now, but discuss this issue in Section 5. For now we will concentrate on the problem of computing $\overline{F}$.

2.2. *Computing the Perturbed Function $\overline{F}$*

First we have to settle on the model of computation—what kind of algorithms do we consider? Most algorithms in computational geometry can be modeled by so-called *extended algebraic decision trees* [15]. These are ternary trees, where each interior node v is labeled by a test function $f_v\colon \mathbb{R}^N \mapsto \mathbb{R}$ and its branches labeled -1, 0, and $+1$, respectively; each leaf v is labeled with a result function $r_v\colon \mathbb{R}^N \mapsto \mathcal{O}$ (we are now assuming that $\mathcal{I} = \mathbb{R}^N$). Computation with such a tree works as follows: upon input $q \in \mathbb{R}^N$ the function sign $f_v(q)$ is evaluated, where v is the root node; the branch labeled by the outcome of this evaluation is taken, and the computation is continued in that subtree; if a leaf ℓ is reached, then $r_\ell(q)$ is evaluated and returned as the "output" of the computation. It is assumed here that the test functions f_v and the result functions r_ℓ are defined and continuous for each q that "reaches" their node v or ℓ. Moreover, each test function f_v must be "easily computable." This is the case, for instance, if it is a small degree polynomial.

There is a slightly more powerful model, called *algebraic computation trees* [15], which we will not consider in this paper. Extended algebraic decision trees abstract away all bookkeeping and storage details. They model many geometric algorithms well, since such algorithms often rely exclusively on a small set of so-called geometric primitive

tests (see [11]) for their geometric content. These tests supply the test functions f_v in the tree model.

For the planar convex hull examples, typical primitives will be coordinate comparisons, $x_i \lessgtr x_j$ or $y_i \lessgtr y_j$, yielding test functions $f_v(q) = x_i - x_j$ and $f_v(q) = y_i - y_j$, and so-called sidedness tests, that determine the relative orientation of three points (x_i, y_i), (x_j, y_j), (x_k, y_k), in particular whether they are collinear; this is expressed by the test function

$$f_v(q) = \begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{vmatrix}.$$

Other geometric primitives tests that typically arise are so-called in-circle or in-sphere tests, the higher-dimensional version of the sidedness test, distance comparisons, and others. They can all be expressed by suitable, relatively simple test functions f_v.

Now assume we have an extended algebraic decision tree T that computes some function $F\colon \mathbb{R}^N \mapsto \mathcal{O}$. How can we compute the perturbed function $\overline{F}^Q$ for some perturbation scheme Q? It is easy to see that all we need to do is the following *perturbed evaluation of* T: at each internal node v, instead of $s_v(q) = \text{sign } f_v(q)$, evaluate the "perturbed test function" $\overline{s_v}^Q(q) = \lim_{\varepsilon \to 0^+} \text{sign } f_v(q(\varepsilon))$; for each leaf ℓ compute and return $\overline{r_\ell}^Q(q) = \lim_{\varepsilon \to 0^+} r_\ell(q(\varepsilon))$.

Let us ignore for the time being how the new perturbed test and result functions involving the limits can be evaluated.

Definition 4. Let $f\colon \mathbb{R}^N \mapsto \mathbb{R}$ be a function, let $q \in \mathbb{R}^N$, and let $q(\,)$ be a perturbation (curve) of q. We say that $q(\,)$ is *valid for* f iff $\lim_{\varepsilon \to 0^+} \text{sign } f(q(\varepsilon))$ exists and is not zero. A perturbation scheme Q is valid for f iff $q(\,)$ is valid for f for each $q \in \mathbb{R}^N$. If $\mathcal{F}$ is a family of functions f, then we say that a perturbation or a perturbations scheme is valid for $\mathcal{F}$ if it is valid for each $f \in \mathcal{F}$.

Let T be a tree for computing F as above and let Q be a perturbation scheme that is valid for $\mathcal{F}$, the set of all test functions f_v appearing in that tree. If we now use the method outlined above for using T to compute the perturbed function $\overline{F}^Q$, then, as is easy to see, no branch labeled 0 will ever be followed during a computation. Thus these branches could all be pruned away. Even better, if T was incomplete and had some of the 0-branches missing, it still can be used to compute $\overline{F}^Q$. And this is the observation that lies at the heart of the whole perturbation method. Even if we have an incomplete program that computes F for most q but misses parts that deal with inputs q that make some geometric primitive tests evaluate to 0 (and are hence deemed "degenerate"), we can still make this program compute $\overline{F}^Q$ for all q. This is all expressed by the following theorem:

Theorem 5. *Let T be a correct extended algebraic decision tree computing some function $F\colon \mathbb{R}^N \mapsto \mathcal{O}$, and let Q be a perturbation scheme that is valid for the set of test functions appearing in T.*

1. *A perturbed evaluation of T computes the perturbed function $\overline{F}^Q$.*

2. *If F is continuous at q, then the perturbed evaluation of T with input q yields $F(q)$.*
3. *The above statements remain true, if some, or all, of the* 0*-branches of T are removed.*

Note one interesting fact: The method prescribes that the perturbed test function $\overline{s_v}^Q$ has to be evaluated for *all* v along the computation path. This means that if we have a program that has some "degenerate cases" programmed and others missing, and we introduce the perturbation method, then we have to discard the programmed "degenerate" cases (unless there is some additional information and reasoning available)! This may seem counterintuitive, but here is an example situation that can arise in a planar convex hull computation. Say we have a program that has dealt with all three cases for coordinate comparisons, but only with the two "nondegenerate" cases for the sidedness tests, and we apply perturbation only to the sidedness tests. If now three input points lie on a common vertical line, then the program might detect their collinearity because of coordinate comparisons, but also detect noncollinearity because of a perturbed sidedness test. Needless to say, this can lead to severe consistency problems within the program.

Furthermore, note that the assumption of Theorem 5 of the validity of the perturbation scheme Q for each test function f appearing in T implies that no f must be the zero function: $f(x) = 0$ for all x implies that $\lim_{\varepsilon\to 0^+} \operatorname{sign} f(q(\varepsilon)) = 0$. However, it is conceivable that an all zero test function appears in a decision tree corresponding to a geometric algorithm. For instance, an algorithm may contain a sidedness test comparing planar point a with the straight line defined by a and b. Of course a always lies on that line and this fact cannot be removed through perturbation tricks. This means that there are special "degenerate cases" that cannot be dealt with automatically using any perturbation method and they have to be taken care of by hand.

In order to apply the perturbation method to some incomplete program we need three things: (1) a valid perturbation scheme, or at least a way to come up with a valid perturbation for each input; (2) a way to evaluate the perturbed test functions; and (3) a way to evaluate the perturbed result functions $\overline{r_\ell}^Q$.

Points (1) and (2) we will deal with in the following sections. Regarding (3) we will just offer a few comments. Let ℓ be a leaf of T that is reached from the root by following only nonzero branches. Let X_ℓ be the set of inputs q that reach ℓ upon normal evaluation of T, and let $\overline{X_\ell}$ be the set of inputs that reach ℓ upon perturbed evaluation. Assuming reasonably behaved test functions, X_ℓ will be an open subset of $\mathbb{R}^N$, and $\overline{X_\ell}$ will contain X_ℓ plus some boundary points. (Although the notation may suggest it, $\overline{X_\ell}$ will in general not be the topological closure of X_ℓ.) Perturbed evaluation of T requires the evaluation of the perturbed result function $\overline{r_\ell}$ for $q \in \overline{X_\ell}$. This is no problem if $q \in X_\ell$, since by assumption r_ℓ is continuous for such q, and hence $\overline{r_\ell}(q) = r_\ell(q)$. Problems can only arise with boundary points; for instance, $\overline{r_\ell}(q)$ might not even be defined for such points q (although this means that $\overline{F}(q)$ is not defined either). However, in many cases there will not be a problem at all. It may be that r_ℓ is constant for all $q \in X_\ell$, and hence $\overline{r_\ell}(q)$ is the same constant for boundary points q. This happens, for instance, in the convex hull sequence problem. It may also happen that the expression computing r_ℓ actually yields a function that is continuous for all of $\mathbb{R}^N$, in which case we have $\overline{r_\ell}(q) = r_\ell(q)$.

This happens, for instance, in the convex hull area problem, where r_ℓ will typically be the sum of the areas of triangles or trapezoids spanned by various input points. At this point the astute reader will certainly have noticed that the result functions r_ℓ are one of the weak spots of the extended algebraic decision tree model.

3. Linear Perturbations

A perturbation $q(\,)$ of a point $q \in \mathbb{R}^N$ is called *linear* if $q(\varepsilon) = q + \varepsilon a_q$, where a_q is some direction, i.e., nonzero vector in $\mathbb{R}^N$. A perturbation scheme Q is called *linear* if $q(\,)$ is linear for each q.

Linear perturbations are interesting because they tend to allow relatively easy evaluation of perturbed functions.

Theorem 6. *Let $f\colon \mathbb{R}^N \mapsto \mathbb{R}$ be a multivariate polynomial of total degree at most Δ, and let B_f be a "black box algorithm" computing f. Let $q(\,)$ be a linear perturbation of q that is valid for f. Then*

$$\lim_{\varepsilon \to 0^+} \operatorname{sign} f(q(\varepsilon))$$

can be determined using at most $\Delta + 1$ calls to B_f plus some small overhead.

Proof. Since $q(\,)$ is linear, $f(q(\varepsilon))$ is a polynomial $p(\varepsilon)$ of degree at most Δ. The desired limit is given by the sign of the smallest degree nonzero coefficient of p (which exists because of the validity of the perturbation). But the coefficients of p can be determined by first computing $p(\varepsilon) = f(q + \varepsilon a_q)$ for $\varepsilon = 0, 1, \ldots, \Delta$ using the black box B_f, and then using polynomial interpolation. Using a naive method this interpolation incurs an overhead of $O(\Delta^3)$ time.

Note that if $p(0)$ evaluates to nonzero, nothing more needs to be done, since it is the smallest degree coefficient of p. □

A similar theorem can be proven if f is, say, the quotient of two multivariate polynomials of bounded degree.

This theorem makes the perturbation method readily applicable to a large class of programs computing geometric functions. Almost all geometric primitive tests seem to be expressible by test functions that are small degree multivariate polynomials. This is certainly true for all those mentioned before, such as coordinate comparisons, sidedness tests, in-sphere tests, and distance comparisons. We are still not home free, though, since the theorem makes the assumption that a valid linear perturbation is available. How can we obtain such a perturbation?

3.1. *A Random Construction*

Let us first consider the geometric meaning of validity. Let f be some continuous real valued function on $\mathbb{R}^N$. Then $f^{-1}(0)$, the zero set of f, forms a surface σ in $\mathbb{R}^N$. Let

$q(\,)$ be some curve starting at the point q. If some open initial segment of $q(\,)$ does not intersect σ, i.e., $f(q(\varepsilon)) \neq 0$ for $0 < \varepsilon < \varepsilon_1$ for some $\varepsilon_1 > 0$, then for all points $x = q(\varepsilon)$ on this segment $f(x)$ must have the same nonzero sign s. This follows from applying the mean value theorem to the continuous function $g(\varepsilon) = f(q(\varepsilon))$. But this immediately implies that $\lim_{\varepsilon \to 0^+} \operatorname{sign} f(q(\varepsilon)) = s$, i.e., the limit exists and is nonzero. In other words, if some open initial segment of the curve $q(\,)$ fails to intersect the surface σ, then $q(\,)$ is valid for f.

Now let $q(\,)$ be a linear perturbation, i.e., a line, or more precisely, a ray starting at some point q, and let σ be some surface. How could $q(\,)$ not be valid, i.e., how could every open initial segment of this ray intersect σ? Certainly there is no problem if q does not lie in σ. Otherwise, if q lies in σ, validity may fail because either: (1) σ contains some entire initial segment or even all of the ray; or (2) σ is very badly behaved (e.g., some analogue of a space-filling curve, or of $x \sin(1/x)$). Let us ignore (2) and assume that σ is well behaved, in a sense "smooth." (For practically all interesting functions f, such as for instance polynomials, $\sigma = f^{-1}(0)$ is "smooth.") In that case, "most" rays starting in q will be okay; the set A of all directions a so that the perturbation $q(\varepsilon) = q + \varepsilon a$ is not valid will be of measure 0. This means that if we pick a direction a from $\mathbb{R}^N$ at random, with probability 1 the resulting linear perturbation of q will be valid for σ. It follows that this "random" linear perturbation of q is also valid with probability 1 for any finite set Σ of smooth surfaces (or any finite set $\mathcal{F}$ of functions whose zero sets form smooth surfaces).

This method of selecting a valid perturbation is less attractive than it looks at first, since it is not clear how to randomly select a random direction a from $\mathbb{R}^N$ within reasonable time. So the question arises whether one can choose a randomly from a much smaller, finite set. This is possible and was first analyzed in [9] for polynomial and rational surfaces of bounded degree. We present a somewhat modified version here, just dealing with the polynomial case. The rational case is similar.

Let T be some extended algebraic decision tree, where each test function is a multivariate polynomial of total degree at most d. Let $\mathcal{F}$ be the set of test functions in T, and let $S = |\mathcal{F}|$. Certainly $S \leq 3^t$, where t is the height of T (in other words, the worst-case running time of the program modeled by T). In many cases, however, S will be much smaller; for instance, for a tree modeling a typical planar convex hull algorithm $S = O(n^3)$, since n planar points allow only these many different sidedness tests. Let $[m]$ denote $\{1, 2, \ldots, m\}$. Let $q \in \mathbb{R}^N$ be some fixed input to T. The question now is: What is the probability that for a direction a chosen from $[m]^N$ uniformly at random the perturbation $q(\varepsilon) = q + \varepsilon a$ is valid for all[3] S test functions in $\mathcal{F}$. Moreover, how large do we need to make m so that this probability becomes suitably small?

Theorem 7. *Let T be an extended algebraic decision tree with a set $\mathcal{F}$ of S different test functions, each a multivariate polynomial of total degree at most d, and let $q \in \mathbb{R}^N$ be a fixed input to T. If direction a is chosen uniformly at random from $[m]^N$, then the linear perturbation $q(\varepsilon) = q + \varepsilon a$ fails to be valid with probability at most dS/m.*

[3] One would think that validity for the at most t-test functions encountered by q on its path down the tree during a perturbed evaluation using a would be good enough. Note however, that this path depends on the choice of a.

Proof. If for any $f \in \mathcal{F}$ the probability that some random $a \in [m]^N$ does not yield a linear perturbation of q valid for f is upper bounded by some number p, then the probability of nonvalidity of such a random a for all of $\mathcal{F}$ is upper bounded by $p|\mathcal{F}| = pS$. Now it only remains to show that we can use $p = d/m$.

The perturbation $q(\varepsilon) = q + \varepsilon a$ is not valid for some polynomial f means $f(q(\varepsilon)) = 0$ for all ε, in particular, for $\varepsilon = 1$, i.e., $f(q + a) = 0$ has to happen. The function $g(a) = f(q + a)$ is a multivariate polynomial in a of total degree at most d. Assuming that f does not vanish everywhere, g does not either. For such a g a lemma due to Schwartz [17] now says that if a is chosen uniformly at random from $[m]^N$ the probability that $g(a) = 0$ is at most d/m. □

Of course, if such a randomly chosen a turns out to be bad, i.e., during the perturbed evaluation of T a 0-branch is to be taken, one simply aborts and restarts with a linear perturbation given by a new randomly chosen a.

3.2. *A Deterministic Construction*

Deterministic construction of a valid linear perturbation seems to be rather difficult for the general case. However, for most of the important special cases arising in computational geometry interesting things can be said.

Definition 8. Let $\sigma = f^{-1}(0)$ be the surface formed by the zero set of some continuous function $f\colon \mathbb{R}^N \mapsto \mathbb{R}$.

We call σ and f *well-behaved* iff every straight line L is either contained in σ or every bounded segment of L intersects σ in at most finitely many points.

We call σ and f *scale-invariant* iff $a \in \sigma$ implies $\lambda a \in \sigma$ for all $\lambda \in \mathbb{R}$.

Many functions are well-behaved; for instance, polynomials, rational functions, and even analytic functions.

Practically all primitive test functions arising in geometric algorithms are scale-invariant. The reason can be exemplified by the sidedness test: if applied to three planar points such a test evaluates to zero, then these points must be collinear, and this happens irrespective of the scale of the coordinate system.

Some natural geometric tests are not scale-invariant because they involve constants, for instance, the test whether the triangle spanned by three planar points has area 1. However, by applying homogenization such a test can usually be transformed into a scale-invariant one, as in our example, where we can replace the constant 1 by the expression u^2, and make u another parameter of the test function.

Lemma 9. *Let $f\colon \mathbb{R}^N \mapsto \mathbb{R}$ be continuous, well-behaved, and scale-invariant, and let a be a direction in $\mathbb{R}^N$ such that $f(a) \neq 0$.*

Then for every $q \in \mathbb{R}^N$ the linear perturbation $q(\varepsilon) = q + \varepsilon a$ is valid for f.

Proof. If $f(q) \neq 0$, then every linear perturbation of q is valid for f, in particular, the one with direction a. So assume $f(q) = 0$. Since f is well-behaved, validity of the

perturbation $q(\varepsilon) = q + \varepsilon a$ can only fail because the line spanned by the perturbation is contained in the zero-surface of f, i.e., $f(q + \varepsilon a) = 0$ for all $\varepsilon \in \mathbb{R}$. But this cannot happen. For in that case scale-invariance of f implies that $f(\lambda(q + \varepsilon a)) = 0$ for all λ, in particular for $\lambda = 1/\varepsilon$, i.e., we would have $f((1/\varepsilon)q + a) = 0$ for all ε. But taking the limit as $\varepsilon \to \infty$ and appealing to the continuity of f we would get $f(a) = 0$, contradicting the assumption $f(a) \neq 0$. □

Rote [16] has pointed out that the assumption of scale-invariance can be dropped.

Lemma 10. *Let $f\colon \mathbb{R}^N \mapsto \mathbb{R}$ be continuous and well-behaved, and let a be a point in $\mathbb{R}^N$ such that $f(a) \neq 0$. Then for every $q \in \mathbb{R}^N$, the linear perturbation $q(\varepsilon) = q + \varepsilon(a - q)$ is valid for f.*

Proof. The line described by $q(\varepsilon)$ contains at least one point that is not in the zero-set of f, namely the point $q(1) = a$. Thus the well-behavedness and continuity of f imply that $\lim_{\varepsilon \to 0^+} \operatorname{sign} f(q(\varepsilon))$ exists and is not zero, i.e., $q(\ )$ is a valid perturbation, as claimed. □

These lemmas immediately yield the following important theorem.

Theorem 11. *Let T be an extended algebraic decision tree algorithm and let $\mathcal{F}$ be its set of test functions. Assume that all $f \in \mathcal{F}$ are well-behaved.*

1. *If all $f \in \mathcal{F}$ are scale-invariant and if a is a direction in $\mathbb{R}^N$ such that $f(a) \neq 0$ for all $f \in \mathcal{F}$, then for* every *$q \in \mathbb{R}^N$ the perturbation $q(\varepsilon) = q + \varepsilon a$ is valid for $\mathcal{F}$.*
2. *If a is a point in $\mathbb{R}^N$ such that $f(a) \neq 0$ for all $f \in \mathcal{F}$, then for* every *$q \in \mathbb{R}^N$ the perturbation $q(\varepsilon) = q + \varepsilon(a - q)$ is valid for $\mathcal{F}$.*

This theorem can be paraphrased as *"if you know just one nondegenerate input, then you can use it for a valid linear perturbation for* every *possible input."*

As already mentioned, the set $\mathcal{F}$ of test functions that appear in algorithms in computational geometry is usually quite limited. An algorithm computing some function on n points in $\mathbb{R}^d$ will typically use geometric primitives such as the $d\binom{n}{2}$ coordinate comparisons, the $\binom{n}{d+1}$ sidedness tests, the $\binom{n}{d+2}$ in-sphere tests, and maybe the $O(n^4)$ possible interpoint distance comparisons. In order to apply our theorem, we now need to come up with a set S of n points in $\mathbb{R}^d$ that are nondegenerate with respect to these primitives, i.e., (1) no two points in S agree in any coordinate; (2) no $d + 1$ points in S lie on the same hyperplane; (3) no $d + 2$ points lie on the same sphere; and (4) all the interpoint distances are distinct.

If we choose n points from the positive branch (i.e., $t > 0$) of the moment curve $\gamma(t) = (t, t^2, \ldots, t^d)$, then (1) is obviously satisfied and because of the nonvanishing of the Vandermonde determinant (2) is satisfied. Using Descartes' Rule of Sign for polynomials it is easy to show that the positive branch of the moment curve intersects no sphere in more than $d + 1$ points; thus choosing the n points from the positive branch ensures (3) also holds. Whether (4) is also satisfied clearly depends on which points are

actually chosen from $\gamma(\ )$. This becomes particularly challenging if one insists on points with integral coordinates. It may be that $\{\gamma(1), \gamma(2), \ldots, \gamma(n)\}$ actually satisfies (4) but I have been unable to prove it. (The fact that for $d = 2$ the points $\gamma(20)$ and $\gamma(-16)$ are equidistant from $\gamma(18)$ provides some negative evidence.) However, note that k points on $\gamma(\ )$ can make at worst the first k^2 integral places on $\gamma(\ )$ ineligible for placing a $(k+1)$st point. Performing a greedy construction this implies that there always is a set I of n integers between 0 and n^2 such that $S = \{\gamma(i) | i \in I\}$ satisfies (4) besides (1), (2), and (3).

Although the moment curve construction yields nondegenerate point sets for the most commonly occurring geometric primitive test functions it is certainly not a panacea that can be applied as easily to other primitives. Moreover, even for the test functions considered here it is not completely satisfying, since integral points are used with rather large coordinates. The question naturally arises how small an integer m we can use so that the grid $\{1, 2, \ldots, m\}^d$ contains a set S of n points nondegenerate with respect to various geometric primitives.

Let $\gamma_p(t) = (t \bmod p, t^2 \bmod p, \ldots, t^d \bmod p)$ a "modular" moment curve, where p is prime. If we use $S = \{\gamma_p(1), \ldots, \gamma_p(n)\}$ it is not too hard to see that this will satisfy (2) if $p > n$, and hence we get $m = O(n)$. This choice will not necessarily satisfy the disjoint coordinate condition (1), since $t^j \bmod p = c$ will have up to j solutions. However, by eliminating points from S that duplicate coordinates, we can obtain a $1/d!$ fraction of the points in S that also satisfy (1), and thus we get $m = O(d!\, n)$ for that case. Thiele [19] has shown that by choosing $p = \Omega(n^{d-1})$ this set S can be made to satisfy the noncosphericity condition (3) also, thus obtaining $m = O(n^{d-1})$ for that case.

Explicitly constructing "nondegenerate" point sets seems to be a difficult problem. In general, it is a special case of the so-called zero avoidance problem of computational algebra.

4. Previous Work

The perturbation method has been used for quite a while in the sense that for specific algorithms *ad hoc* perturbation schemes were proposed. Maybe the first instance was the perturbation of the right-hand side vector in the simplex algorithm proposed by Dantzig [5]. Examples in the computational geometry literature are: the Euclidean maximum spanning tree algorithm in [14], where one needed to avoid duplication among interpoint distances; the general shelling algorithm for computing convex hulls in [18], where one needed to avoid an inadmissible shelling line; and the polyhedron intersection algorithm in [6], where one needed to avoid all kinds of incidences and parallelism between the faces of the polyhedra.

More or less general perturbation methods have been proposed and discussed *per se* by four different groups of authors. Edelsbrunner and Mücke with their SoS ("Simulation Of Simplicity") scheme [7]; Yap with his general symbolic scheme [20], [21]; Canny and Emiris with their efficient linear scheme [9], [8], [4]; and Michelucci with his random scheme [13]. We will briefly review each scheme, and in particular discuss: what kind of perturbation curves are used; for what kind of test functions is the scheme valid for; and how are the perturbed test functions evaluated.

4.1. *The SoS Scheme of Edelsbrunner and Mücke*

This work concentrates mainly on the sidedness test for $d+1$ among n points $p_1, \ldots, p_n$ in $\mathbb{R}^d$, and on related tests that can be expressed by determinants of matrices whose entries are coordinates of the points.

The perturbation scheme uses polynomial curves of high degree that vary $p_{i,j}$, the jth coordinate of p_i, as

$$p_{i,j}(\varepsilon) = p_{i,j} + \varepsilon^{2^{i \cdot d - j}}.$$

The determinant of a matrix $A(\varepsilon)$ involving such entities is then an extremely high degree polynomial $P(\varepsilon)$.

Evaluation of such a perturbed test now means evaluating $\lim_{\varepsilon \to 0^+} P(\varepsilon)$, which amounts to finding the smallest degree nonzero coefficient of P. These coefficients turn out to be determinants of certain submatrices of $A(0)$. Slightly involved analysis then shows how these submatrices can be generated in the desired order.

4.2. *The Linear Scheme of Canny and Emiris*

The main emphasis of this work by Canny and Emiris lies in the fast evaluation of perturbed test functions. They proposed the use of linear perturbation, and proved the validity of the specific perturbation given by

$$p_{i,j}(\varepsilon) = p_{i,j} + \varepsilon \cdot i^j$$

for the coordinate comparison, sidedness, and in-sphere test functions mentioned above. (Of course, this is precisely the perturbation induced by directions from the moment curve described at the end of the previous section. Directions stemming from the modular moment curve were also considered by those authors.) For the perturbed sidedness function they noticed that evaluation really amounted to determining the coefficients of the polynomial given by the determinant $\det(A + \varepsilon B)$, where A and B are matrices, and that this polynomial is essentially the same as the characteristic polynomial of $A \cdot B^{-1}$, which can be computed very quickly. They also showed that the evaluation of the perturbed in-sphere function could be reduced to some characteristic polynomial computation.

Along more general lines they proved a version of Theorem 7, thus showing that using random choice linear perturbations could be found that are valid for finite sets of bounded degree polynomial and rational functions. For the evaluation of the perturbed version of such functions they seem to have overlooked the interpolation-based Theorem 6. Instead they assumed that programs were available for the original function which, by replacing arithmetic on reals with arithmetic on ε-polynomials and rationals, could be transformed into programs for the perturbed functions.

The running times of all the various evaluations were very carefully analyzed under different cost models of computation.

4.3. *The Random Scheme of Michelucci*

This scheme has the components of the perturbation curves given by randomly chosen power series. Computation is then performed on those power series instead of on the original variables. In order to avoid computing with an infinite object such as a complete power series, only an initial portion of the coefficients are generated lazily in a piecemeal fashion and used. With probability 1 finite initial portions of such a power series will suffice in order to avoid 0-branches of sign tests.

This scheme has the advantage that it quite naturally also works in the algebraic computation tree model, and not just the decision tree model. However, the programming effort needed to deal lazily with such possibly expanding initial portions of power series is only reasonably achievable if relatively sophisticated language constructs such as streams are available. In addition, the overhead incurred in terms of time and space seems formidable.

4.4. *The Symbolic Scheme of Yap*

Yap's scheme is the most general, and in a way the most intriguing. It applies to arbitrary polynomial and rational test functions, and, as we will see shortly, even to analytic functions. Interestingly, it does not specify perturbation curves at all. Instead, it just gives a method that for any function f and any q produces a sign $s \in \{-1, +1\}$ such that $s = \operatorname{sign} f(q)$ if $f(q) \neq 0$. The "difficulty" with this scheme is understanding why it really works, why it is not like an arbitrary sign assignment, and why no inconsistencies can occur. Yap offers two proofs for the consistency of his scheme, an algebraic proof based on the theory of sign functions [20], and a highly technical geometric proof, in which perturbations are understood as converging point sequences [21]. After describing Yap's method we will give a rather simple argument based on our framework showing that Yap's symbolic method behaves as if certain polynomial perturbation curves were used.

Let us establish some notation. For $x, y \in \mathbb{R}^n$ we let $\langle x, y\rangle$ denote the inner product $\sum_{1\le i\le n} x_i y_i$. Recall the identity $\langle x, Wy\rangle = \langle W^T x, y\rangle$, where W is an $n \times n$ matrix and W^T its transpose. For $x, y \in \mathbb{R}^n$ we let x^y denote the scalar $x_1^{y_1} \cdot x_2^{y_2} \cdots x_n^{y_n}$, and for $\lambda \in \mathbb{R}$ we let λ^x denote the vector $(\lambda^{x_1}, \lambda^{x_2}, \ldots, \lambda^{x_n})$. Thus if $a \in \mathbb{R}^n$ and if $\overline{n}$ denotes $(n, n-1, \ldots, 1)$, then $\langle a, \lambda^{\overline{n}}\rangle$ is just strange notation for the polynomial $a_1\lambda^n + a_2\lambda^{n-1} + \cdots + a_n\lambda^1$. For $\mu \in \mathbb{N}^n$ we let $\mu!$ denote $\mu_1!\,\mu_2! \cdots \mu_n!$, and for a function $f\colon \mathbb{R}^n \mapsto \mathbb{R}$ we let $f^{(\mu)}$ stand for the partial derivative $(\partial^{\mu_1}/\partial x_1^{\mu_1})(\partial^{\mu_2}/\partial x_2^{\mu_2}) \cdots (\partial^{\mu_n}/\partial x_n^{\mu_n})\, f$.

Let $<_\ell$ denote the usual lexicographic ordering on $\mathbb{N}^n$. Now any $n \times n$ invertible matrix W induces a total ordering $<_W$ on $\mathbb{N}^n$, where $\mu <_W \nu$ iff $W\mu <_\ell W\nu$. If an invertible W contains only nonnegative integer entries we call $<_W$ an *admissible ordering*.[4] The best-known examples of admissible orderings are $<_I$, where I is the identity matrix, which is of course nothing but the lexicographic ordering, and $<_U$, where U is all zero, except for the first row, which is all 1, and 1's below the main diagonal, i.e., $U_{i,i-1} = 1$

[4] Yap also allows matrices with nonnegative real entries.

for $1 < i \leq n$. In the theory of multivariate polynomials $<_U$ is known as the total degree order.

After fixing an admissible ordering $<_W$, Yap uses the following method, that given a function $f\colon \mathbb{R}^n \mapsto \mathbb{R}$ and a $q \in \mathbb{R}^n$ produces an $s = s(f, q) \in \{-1, +1\}$:

As $s = s(f, q)$ produce sign $f^{(\mu)}(q)$, where μ is the minimum element of $\mathbb{N}^n$ under $<_W$ with $f^{(\mu)}(q) \neq 0$.

Note that since for any admissible ordering the 0-vector is the minimum in $\mathbb{N}^n$, the s produced by this method has the property that if $f(q) \neq 0$, then $s = \operatorname{sign} f(q)$. However, the interesting case is $f(q) = 0$. Is there a perturbation curve $q(\,)$ producing the same sign? i.e., is there a curve $q(\,)$ such that $s(f, q) = \lim_{\varepsilon \to 0^+} \operatorname{sign} f(q(\varepsilon))$ holds? This is indeed the case, if one considers only a single function f or a finite set of such functions.

Theorem 12. *Let $<_W$ be an admissible ordering induced by matrix W. Let $\mathcal{F}$ be a finite set of not everywhere vanishing analytic functions $f\colon \mathbb{R}^n \mapsto \mathbb{R}$, let $q \in \mathbb{R}^n$ be in the domains of those functions, and let $s(f, q)$ be the sign produced by Yap's method.*

For any sufficiently large integer λ the perturbation curve $q(\varepsilon) = q + \varepsilon^{W^T \lambda^{\overline{n}}}$ has the property that for all $f \in \mathcal{F}$ we have $s(f, q) = \lim_{\varepsilon \to 0^+} \operatorname{sign} f(q(\varepsilon))$.

Note. A function $f\colon \mathbb{R}^n \mapsto \mathbb{R}$ is *analytic* if for each point q of its domain f can be represented as a multivariate convergent power series, i.e., $f(x) = \sum_{\mu \in \mathbb{N}^n} a_\mu (x - q)^\mu$ in some neighborhood of q, where $x = (x_1, \ldots, x_n)$ and $a_\mu \in \mathbb{R}$. It is one of the properties of analytic functions that $a_\mu = f^{(\mu)}(q)/\mu!$. Analytic functions are closed under the usual arithmetic operations and under composition. They include polynomials, rational functions, and many more. See [12] for more information.

Proof. Since $\mathcal{F}$ is finite it suffices to prove the claim of the theorem for any $f \in \mathcal{F}$. For such an f we need to determine $\lim_{\varepsilon \to 0^+} \operatorname{sign} f(q + \varepsilon^{W^T \lambda^{\overline{n}}})$.

Using the power series representation $\sum_{\mu \in \mathbb{N}^n} a_\mu (x - q)^\mu$ for f around q we get

$$f(q + \varepsilon^{W^T \lambda^{\overline{n}}}) = \sum_{\mu \in \mathbb{N}^n} a_\mu (\varepsilon^{W^T \lambda^{\overline{n}}})^\mu = \sum_{\mu \in \mathbb{N}^n} a_\mu \varepsilon^{\langle W^T \lambda^{\overline{n}}, \mu \rangle} = \sum_{\mu \in \mathbb{N}^n} a_\mu \varepsilon^{\langle \lambda^{\overline{n}}, W\mu \rangle},$$

i.e., we have represented $f(q(\varepsilon))$ as a (convergent) power series in ε which does not vanish everywhere because f does not. Now if $g(\varepsilon) = \sum_{i \in \mathbb{N}} b_i \varepsilon^i$, then $\lim_{\varepsilon \to 0^+} \operatorname{sign} g(\varepsilon)$ is given by the sign of b_i where i is the smallest index such that $b_i \neq 0$. This means that, since sign $a_\mu = \operatorname{sign} f^{(\mu)}(q)$, we are done if we can show the following: Let $M = \{\nu \in \mathbb{N}^n | a_\nu \neq 0\}$. If μ is the minimum element in M under the order $<_W$, then for any sufficiently large integer λ the number $\langle \lambda^{\overline{n}}, W\mu \rangle$ is smaller than all other numbers in the set $M' = \{\langle \lambda^{\overline{n}}, W\nu \rangle | \nu \in M\}$. But μ is the minimum element in M under $<_W$ means that $\mu' = W\mu$ is the lexicographically minimum element in $\{\nu' = W\nu | \nu \in M\}$. And since $\mu' <_\ell \nu'$ implies $\langle \lambda^{\overline{n}}, \mu' \rangle < \langle \lambda^{\overline{n}}, \nu' \rangle$ for integer λ bigger than the largest coordinate of μ', we get that $\langle \lambda^{\overline{n}}, \mu' \rangle$ is indeed the smallest in the set M'. □

Note that using the proof just offered one can arrive naturally and automatically at the SoS scheme of Edelsbrunner and Mücke: They consider determinant functions of the input points, which means they are dealing with multivariate polynomials where the maximum degree of any variable is 1. In other words, the exponent vectors μ in the "power series" representations only consist of 0's and 1's. Now use for W the identity matrix. Then the largest coordinate of any $\mu' = W\mu$ in the proof above is 1. The proof now tells us that λ "sufficiently large" is achieved by $\lambda = 2$. Thus $q(\varepsilon) = q + \varepsilon^{W^T 2^{\overline{n}}} = q + \varepsilon^{2^{\overline{n}}}$ is a valid perturbation curve, and one obtains the SoS scheme exactly. The proof also suggests that if one intends to use an SoS-type scheme that is also to work for in-sphere predicates, one really ought to use a perturbation curve of the form $q(\varepsilon) = q + \varepsilon^{3^{\overline{n}}}$.

Yap's scheme is very general and powerful. It has the apparent drawback that we need to be able to evaluate potentially arbitrarily high derivatives of the test functions. This is quite unpalatable if each and every one of these derivatives has to be hand-coded. Canny [3] has suggested the interesting idea of computing and evaluating these derivatives on the fly, only when actually needed. There is actually a sizable literature on automatic program differentiation [10]. The idea is that if a program keeps a trace of its arithmetic operations and builds up a (large) acyclic computation graph encoding the expressions the program has formed so far, then at any point the derivative of any expression with respect to any input variable can be computed using simple chain rules. It turns out that such a system would have only modest time overhead. The space overhead incurred through the computation graph, however, would be rather large.

5. Discussion

Is it really advisable to employ perturbations in geometric computing? In the late 1980s convential wisdom would have said yes. Recently, several objections have been voiced and the answer is now not so clear any more.

I have tried to make clear in this paper, that employing the perturbation method means that instead of computing some function F we are now computing a (possibly) different function $\overline{F}$. This fact creates a number of difficulties.

Since we typically really want F and not $\overline{F}$, we need to recover F from $\overline{F}$ somehow via some sort of postprocessing step. This postprocessing step may actually be more difficult or complicated than evaluating F in the first place. This has been exemplified recently by Burnikel *et al.* [2] with the problem of line segment intersection, and also seems implicit in [6].

Computing $\overline{F}$ may take substantially more time that computing F would require. Here are two typical examples, also given in [2]: (1) n segments in the plane intersecting in one point; a reasonable intersection detection algorithm should be able to deal with such a set in $O(n \log n)$ time; if, however, the segments are perturbed, then there will be $\binom{n}{2}$ intersections and any algorithm will need $\Omega(n^2)$ on this perturbed set. (2) One wishes to compute the convex hull of n copies of the same point in $\mathbb{R}^d$; a reasonable algorithm should be able to do this in $O(n)$ time; if linear perturbation is applied to those points with the directions chosen from the moment curve as in Section 3, then the convex hull of the perturbed set will be a cyclic polytope with $O(n^{\lfloor d/2 \rfloor})$ facets, and any algorithm

will need $\Omega(n^{\lfloor d/2 \rfloor})$ time to compute that. This may be just a trivial example. However, recently, Avis *et al*. [1] have reported large classes of point sets S with the property that if $F(S)$ is the number of facets of the convex hull of S, then $\overline{F}(S)$ is much larger than $F(S)$, *no matter what perturbation scheme is employed.*

We may be content with computing $\overline{F}$ instead of F. However, this may lead to problems when F (or $\overline{F}$) is part of a larger system. Even if for all the functions that appear in this system we use a perturbed version it will be rather difficult to make the perturbations interact in a graceful manner.

But the severest shortcoming of the perturbation method may actually be something that, so far, we have swept under the rug completely. The perturbation method relies on and assumes the availability of exact arithmetic. Whether such an assumption is viable in real-world computing remains to be seen.

Acknowledgments

I would like to acknowledge lengthy discussions with John Canny, Herbert Edelsbrunner, Ioannis Emiris, Jeff Erickson, Kurt Mehlhorn, Chee Yap, and Günter Rote. Finally, my apologies to Richard Dedekind for stealing the title.

References

1. D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms? *Comput. Geom. Theory Appl.* To appear.
2. C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. *Proc.* 5*th Annual ACM–SIAM Symp. on Discrete Algorithms*, 1994.
3. J. Canny. Private communication.
4. J. Canny, I. Emiris, and R. Seidel. Efficient perturbations for handling geometric degeneracies. *Algorithmica*. To appear.
5. G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.
6. K. Dobrindt. Algorithmen für Polyeder. Diplomarbeit, FB 14, Informatik, Universitat des Saarlandes, Saarbrücken, 1990.
7. H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graphics* **9**(1) (1990), 67–104.
8. I. Emiris and J. Canny. An efficient approach to removing geometric degeneracies. *Proc.* 8*th Annual ACM Symp. on Comput. Geom.*, 1991, pp. 74–82.
9. I. Emiris and J. Canny. A general approach to removing degeneracies. *SIAM J. Compt.* **24**(3) (1995), 650–664.
10. A. Griewank and G. F. Corliss. *Automatic Differentiation of Algorithms*: *Theory, Implementation, and Applications*. SIAM, Philadelphia, PA, 1991.
11. L. J. Guibas and J. Stolfi. Primitives for manipulation of general subdivisions and computation of Voronoi diagrams. *ACM Trans. Graphics* **4**(2) (1985), 74–123.
12. S. G. Krantz and H. R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser-Verlag, Boston, 1992.
13. D. Michelucci. An ε-arithmetic for removing degeneracies. *Proc. IEEE Symp. on Comput. Arithmetic*, 1995.
14. C. Monma, M. Paterson, S. Suri, and F. Yao. Computing Euclidean maximum spanning trees. *Proc.* 4*th Annual ACM Symp. on Comput. Geom.*, 1988, pp. 241–251.
15. F. P. Preparata and M. I. Shamos. *Computational Geometry, An Introduction*. Springer-Verlag, New York, 1985.

16. G. Rote. Private communication.
17. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.* **27**(4) (1980), 701–717.
18. R. Seidel. Output-size sensitive algorithms for constructive problems in computational geometry. Ph.D. thesis, Computer Science Department, Cornell University, 1986.
19. T. Thiele. Private communication.
20. C.-K. Yap. Symbolic treatment of geometric degeneracies, *J. Symbolic Comput*. **10** (1990), 349–370.
21. C.-K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. Systems Sci*. **40** (1990), 2–18.

Received October 9, 1995, *and in revised form April* 4, 1996.