Algorithmica (1993) 9: 534–560

Algorithmica
© 1993 Springer-Verlag New York Inc.

# Constructing Strongly Convex Approximate Hulls with Inaccurate Primitives

Leonidas Guibas,[1, 2] David Salesin,[1, 3] and Jorge Stolfi[2]

**Abstract.** The first half of this paper introduces *Epsilon Geometry*, a framework for the development of robust geometric algorithms using inaccurate primitives. Epsilon Geometry is based on a very general model of imprecise computations, which includes floating-point and rounded-integer arithmetic as special cases. The second half of the paper introduces the notion of a $(-\varepsilon)$-*convex polygon*, a polygon that remains convex even if its vertices are all arbitrarily displaced by a distance of $\varepsilon$ of less, and proves some interesting properties of such polygons. In particular, we prove that for every point set there exists a $(-\varepsilon)$-convex polygon $H$ such that every point is at most $4\varepsilon$ away from $H$. Using the tools of Epsilon Geometry, we develop robust algorithms for testing whether a polygon is $(-\varepsilon)$-convex, for testing whether a point is inside a $(-\varepsilon)$-convex polygon, and for computing a $(-\varepsilon)$-convex approximate hull for a set of points.

**Key Words.** Computational geometry, Epsilon Geometry, Approximate computations, Robust algorithms, Strongly convex polygons, Convex hull.

## 1. Introduction.

Finding the convex hull of a set of points is one of the simplest and oldest problems in computational geometry, and yet is still a surprisingly difficult problem to solve in practice. The major difficulty is that the basic geometric tests needed to solve the problem are unreliable or inconclusive when implemented with imprecise computations, such as ordinary floating-point arithmetic. This uncertainty makes it extremely difficult to construct the correct hull, the smallest polygon that is simultaneously convex and that contains every point of the given set.

The main result of this paper is a proof that for every point set there exists a polygon that is convex enough to be found with approximate tests, and that is also very close to containing all the points of the given set. In addition, we present an algorithm for finding such hulls that uses inaccurate primitives and that runs in $O(n^3 \log n)$ time in the worst case, and in $O(n \log n)$ expected time for points uniformly distributed in a square. We also give an $O(\log n)$-time algorithm for testing whether or not a point lies inside one of these hulls, again using only approximate tests.

The development of robust geometric algorithms has been attracting increasing attention in recent years [2], [3], [5], [9]–[19], [21]–[23]. Knuth [13], for

[1] Computer Science Department, Stanford University, Stanford, CA 94305, USA.
[2] DEC Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, USA.
[3] Current address: Program of Computer Graphics, Cornell University, Ithaca, NY 14853, USA.

Received November 6, 1990; revised December 19, 1991. Communicated by Takao Asano.

example, describes an algorithm for computing the convex hull of a set of points whose coordinates have been rounded to a sufficiently small number of bits to allow the outcomes of the underlying geometric tests to be determined exactly using fixed- or floating-point arithmetic.

Our algorithm, however, more closely resembles those of Fortune [3], Jaromczyk and Wasilkowski [12], and Milenkovic and Li [18], in that all geometric computations are assumed to be imprecise by nature. The approach in all of these works is to compute an exact result for a perturbed version of the input.

Fortune [3] describes an $O(n \log n)$-time algorithm for computing approximate convex hulls using floating-point arithmetic. One drawback of this algorithm is that the resulting hulls are only approximately convex, and therefore do not enjoy many of the nice properties associated with convexity. By contrast, the hulls computed by our algorithm are not only convex, but so convex that many of these desirable properties are preserved in some fashion even when they are tested with floating-point arithmetic.

Jaromczyk and Wasilkowski [12] present an $O(n)$-time algorithm for constructing the convex hull of a simple polygon, using floating-point arithmetic. As in the previous work, the algorithm has the drawback that the resulting hulls are only approximately convex. In addition, the algorithm requires that the input polygon be sufficiently "well conditioned."

Recently, Milenkovic and Li [18] have also presented results similar to ours. While their algorithm for constructing approximate hulls has a much better worst-case performance than the one presented here, the existence proof and algorithm in this paper establish tighter bounds on the degree of approximation achievable. Moreover, their algorithm assumes certain properties of floating-point arithmetic, whereas the results presented here are based on a much more general model of imprecise computations.

In developing our algorithms, we use the *Epsilon Geometry* framework [7], [21], which is described in Section 2. Epsilon Geometry provides a methodology for developing, proving, and implementing geometric algorithms based on approximate primitives.

## 2. Epsilon Geometry

*2.1. Definitions.* The Epsilon Geometry framework defines the notion of an *epsilon predicate* as a means for expressing "approximate tests" in a general setting. An epsilon predicate is defined as follows:

Let $\mathcal{O}$ be a set of geometric objects endowed with some distance metric $\| \ \|$. Let $P$ be a predicate defined on $\mathcal{O}$. Then, for any $X \in \mathcal{O}$ and any $\varepsilon \geq 0$, we define $\varepsilon$-$P(X)$ as a shorthand for "$P(X')$ is true for *some* $X' \in \mathcal{O}$ such that $\|X, X'\| \leq \varepsilon$." That is, $X$ is at most $\varepsilon$ away from satisfying $P(X)$. The truth-set of $\varepsilon$-$P$, therefore, is that of $P$, "fattened" by $\varepsilon$ (Figure 1). Note that $0$-$P(X)$ is the same as $P(X)$.

In the case of an $n$-ary predicate $P$, we define $\varepsilon$-$P(X_0, \ldots, X_{n-1})$ to mean that $P(X'_0, \ldots, X'_{n-1})$ is true for some $X'_0, \ldots, X'_{n-1}$ with $\|X_i, X'_i\| \leq \varepsilon$ for all $i$.

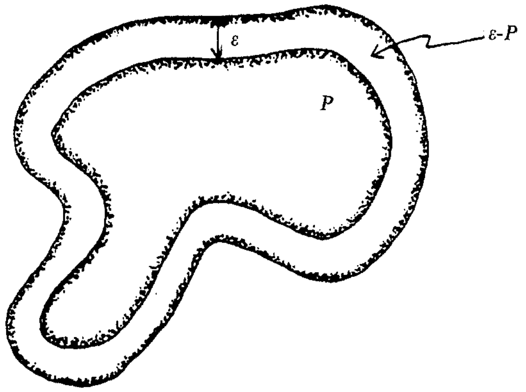

**Fig. 1.** The truth sets of $P$ and $\varepsilon$-$P$.

The following two monotonicity properties follow immediately from the definition of $\varepsilon$-$P$:

$$\begin{aligned} &\varepsilon\text{-}P(X) &\Rightarrow\quad &\varepsilon'\text{-}P(X) &&\text{for all}\quad \varepsilon' \geq \varepsilon, \\ (1)\qquad &\texttt{not}(\varepsilon\text{-}P(X)) &\Rightarrow\quad &\texttt{not}(\varepsilon'\text{-}P(X)) &&\text{for all}\quad \varepsilon' \leq \varepsilon. \end{aligned}$$

We can extend the definition of $\varepsilon$-predicate to negative values of $\varepsilon$ in such a way that these two monotonicity properties remain true for all $\varepsilon$. If $\varepsilon > 0$, we can define $(-\varepsilon)$-$P(X)$ as a shorthand for "$P(X')$ is true for *all* $X' \in \mathcal{O}$ such that $\|X, X'\| \leq \varepsilon$." The truth set of $(-\varepsilon)$-$P$ for $\varepsilon \geq 0$, therefore, is that of $P$, "trimmed" by $\varepsilon$ (Figure 2). This definition can also be expressed by the identity

$$(2)\qquad (-\varepsilon)\text{-}P(X) \quad\Leftrightarrow\quad \texttt{not}(\varepsilon\text{-}(\texttt{not}\, P(X))).$$

Intuitively, an object $X$ that is $(-\varepsilon)$-$P$ is "extremely $P$," whereas an $X$ that is $\varepsilon$-$P$ is only "almost $P$."

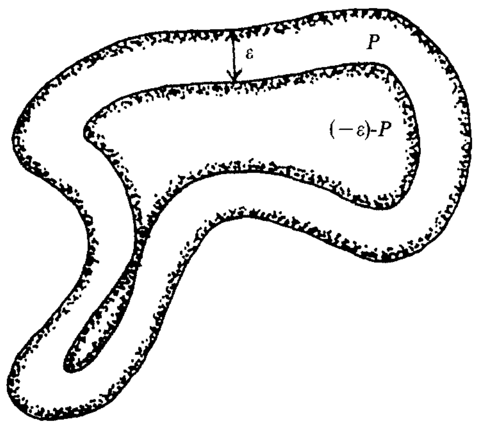

**Fig. 2.** The truth sets of $P$ and $(-\varepsilon)$-$P$.

The following lemmas describe some important properties of epsilon predicates:

Lemma 1. *For any predicates, P and Q, and any $\varepsilon \geq 0$,*

$$\varepsilon\text{-}(P \vee Q)(X) \;\Leftrightarrow\; \varepsilon\text{-}P(X) \vee \varepsilon\text{-}Q(X),$$
$$\varepsilon\text{-}(P \wedge Q)(X) \;\Rightarrow\; \varepsilon\text{-}P(X) \wedge \varepsilon\text{-}Q(X).$$

Proof. For $\varepsilon \geq 0$, the expression $\varepsilon\text{-}(P \vee Q)(X)$ means

$$\exists X'\colon \|X', X\| \leq \varepsilon \wedge (P \vee Q)(X'),$$

which is equivalent to

$$(\exists X'\colon \|X', X\| \leq \varepsilon \wedge P(X')) \vee (\exists X'\colon \|X', X\| \leq \varepsilon \wedge Q(X')),$$

that is, $\varepsilon\text{-}P(X) \vee \varepsilon\text{-}Q(X)$.

Similarly, for $\varepsilon \geq 0$, the expression $\varepsilon\text{-}(P \wedge Q)(X)$ means

$$\exists X'\colon \|X', X\| \leq \varepsilon \wedge (P \wedge Q)(X'),$$

which implies

$$(\exists X'\colon \|X', X\| \leq \varepsilon \wedge P(X')) \wedge (\exists X'\colon \|X', X\| \leq \varepsilon \wedge Q(X')),$$

that is, $\varepsilon\text{-}P(X) \wedge \varepsilon\text{-}Q(X)$. □

Note that in the case of $\wedge$ the implication only works in one direction, because even if it is possible to satisfy $P(X)$ and $Q(X)$ separately with $\varepsilon$-perturbations to $X$, it may not always be possible to satisfy both constraints at once (Figure 3).

Lemma 2. *For any predicates, P and Q, and any $\varepsilon \leq 0$,*

$$\varepsilon\text{-}(P \vee Q)(X) \;\Leftarrow\; \varepsilon\text{-}P(X) \vee \varepsilon\text{-}Q(X),$$
$$\varepsilon\text{-}(P \wedge Q)(X) \;\Leftrightarrow\; \varepsilon\text{-}P(X) \wedge \varepsilon\text{-}Q(X).$$

Proof. Follows immediately from Lemma 1 and equation (2). □

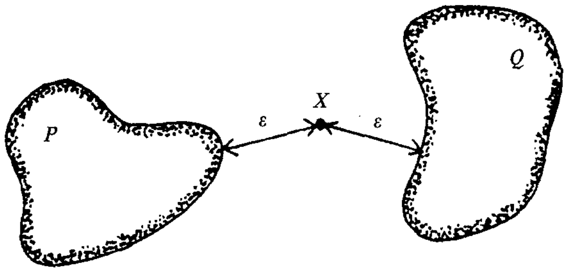

**Fig. 3.** $X$ is $\varepsilon$-$P$ and $\varepsilon$-$Q$ but not $\varepsilon$-$(P \wedge Q)$.

LEMMA 3. *For any $\varepsilon$, $\delta \geq 0$, and any predicates $P$, $Q$, $R$, if $P(X)$ implies $\varepsilon$-$Q(X)$, and $Q(X)$ implies $\delta$-$R(X)$, then $P(X)$ implies $(\varepsilon + \delta)$-$R(X)$.*

PROOF. Suppose that $P(X)$ implies $\varepsilon$-$Q(X)$, $Q(X)$ implies $\delta$-$R(X)$, and $P(X)$ is true. By definition, $\varepsilon$-$Q(X)$ means there exists an object $X'$ such that $\|X, X'\| \leq \varepsilon$ and $Q(X')$ is true. Therefore $\delta$-$R(X')$ is true, so there exists an object $X''$ such that $\|X', X''\| \leq \delta$, and $R(X'')$ is true. By the triangle inequality, $\|X, X''\| \leq \varepsilon + \delta$, which proves $(\varepsilon + \delta)$-$R(X)$ is true. □

We define the *critical perturbation* for a predicate $P$ and an object $X$ to be the unique real value $\bar{\varepsilon}$ such that $\varepsilon$-$P(X)$ is false for $\varepsilon < \bar{\varepsilon}$, and true for $\varepsilon > \bar{\varepsilon}$. (In other words, $\bar{\varepsilon}$ is the distance from $X$ to the closure of the truth set of $P$ if $P(X)$ is false, and minus the distance from $X$ to the closure of the falsehood set of $P$ if $P(X)$ is true.)

*2.2. Implementing Epsilon Predicates.* The epsilon predicates defined above are exact mathematical notions that we use in proofs. In programs, we implement a geometric predicate by a procedure called an *epsilon box*. Typically, an epsilon box will try to evaluate its geometric predicate using floating-point arithmetic or some other approximate methods. An epsilon box may therefore fail to decide whether its predicate is true or false.

We consider here three varieties of epsilon boxes. A *T-box* for a predicate $P(X)$ is a procedure T_P($X$) that computes $P(X)$ and returns either `true`, `false`, or `unknown`. Similarly, an *E-box* for $P(X)$ is a procedure E_P($\varepsilon$, $X$) that computes $\varepsilon$-$P(X)$ and also returns either `true`, `false`, or `unknown`. Thus, T_P($X$) is equivalent to E_P(0, $X$). Finally, an *I-box* for a predicate $P(X)$ is a procedure I_P($X$) that returns an estimate of how far $X$ is from satisfying $P$. The estimate is encoded as an interval $e = (e.lo, e.hi)$ such that $\varepsilon$-$P(X)$ is false for $\varepsilon < e.lo$ and true for $\varepsilon > e.hi$. An interval $e$ with these properties is called an *uncertainty interval* for $P(X)$ (Figure 4). The monotonicity property of epsilon predicates (1) guarantees that we can encode the result of every epsilon-predicate test as a single interval in this manner.

A peculiarity of this encoding is that every I-box must return `unknown` for at least one value of $\varepsilon$, even in cases when it can determine $\varepsilon$-$P(X)$ for all $\varepsilon$ exactly. This problem could be avoided by a more elaborate encoding allowing open and

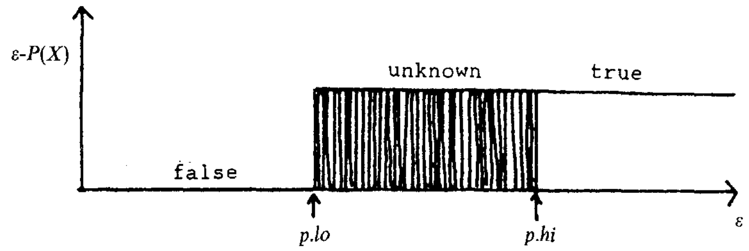

**Fig. 4.** An uncertainty interval for predicate $P(X)$.

half-open uncertainty intervals, but we feel that the infinitesimal gain in accuracy provided by such a scheme would not be worth the added complexity.

Occasionally, we use the notation $e$-$P(X)$ with an interval $e$ to mean `not` $\varepsilon$-$P(X)$ for $\varepsilon < e.lo$ and $\varepsilon$-$P(X)$ for $\varepsilon > e.hi$. Note that the interval $e$ returned by an I-box `I_P`$(X)$ should satisfy $e$-$P(X)$.

*2.3. Accuracy of Epsilon Boxes.* In principle, the Epsilon Geometry framework places only very weak constraints on the outputs of epsilon boxes. An epsilon box (of any variety) is considered *correct* as long as it never returns `false` when the predicate is true, or `true` when the predicate is false. For example, the T-box that always returns `unknown` and the I-box that always returns $(-\infty, +\infty)$ are correct implementations of *any* predicate $P$.

In practice, of course, an epsilon box will be useful only if it can give a definitive `true` or `false` answer for some interesting set of inputs. We say that a T-box `T_P`$(X)$ for a predicate $P(X)$ is *$\lambda$-accurate* (for $\lambda \geq 0$) if for all $X$ the T-box returns `unknown` only when $P(X)$ is at most $\lambda$ away from changing from true to false; that is, only when the critical perturbation $\bar{\varepsilon}$ is at most $\lambda$ in absolute value. Similarly, an E-box `E_P`$(\varepsilon, X)$ is $\lambda$-accurate if it returns `unknown` only when $|\varepsilon - \bar{\varepsilon}| \leq \lambda$. Finally, an I-box `I_P`$(X)$ is $\lambda$-accurate if it always returns an interval $e$ such that $e.lo \geq \bar{\varepsilon} - \lambda$ and $e.hi \leq \bar{\varepsilon} + \lambda$.

We also say, informally, that an epsilon box is *accurate* if it is $\lambda$-accurate for some $\lambda$ small enough to be useful. Ideally, every epsilon-box implemented with floating-point arithmetic should be $\lambda$-accurate, with $\lambda$ equal to a small constant times the machine's floating-point precision. (Our definitions of correctness and accuracy are roughly analogous to Fortune's definitions of "robustness" and "stability" [3].)

*2.4. Rules for Combining Uncertainty Intervals.* In order to construct complex I-boxes out of simpler ones, we must develop techniques for combining the uncertainty intervals that they return. For example, suppose we manage to prove that the predicate $\varepsilon$-$R(X)$ is equivalent to $\varepsilon$-$P(X) \vee \varepsilon$-$Q(X)$, for all $X$ and all $\varepsilon$. Then we could implement `I_R`$(X)$ from `I_P`$(X)$ and `I_Q`$(X)$ as follows:

```
1. $p \leftarrow$ I_P$(X)$.
2. $q \leftarrow$ I_Q$(X)$.
3. $r.lo \leftarrow$ min$\{p.lo, q.lo\}$.
4. $r.hi \leftarrow$ min$\{p.hi, q.hi\}$.
5. return $r$.
```

This code may be easier to understand with the help of Figure 5, which shows the "graphs" of the predicates $\varepsilon$-$P(X)$, $\varepsilon$-$Q(X)$, and $\varepsilon$-$R(X)$ as a function of $\varepsilon$, for a particular $X$. The "fuzzy" portion of each graph represents the uncertainty interval returned by the corresponding I-box.

Note that on most machines the operations required to implement `I_R` can be performed without any rounding errors. Note also that if `I_P` and `I_Q` are $\lambda$-accurate and $\lambda'$-accurate, respectively, then the accuracy of `I_R` is just `max`$\{\lambda, \lambda'\}$. Thus, the I-box for $R$ has the nice property that it is at least as accurate as the least accurate of the I-boxes it uses.

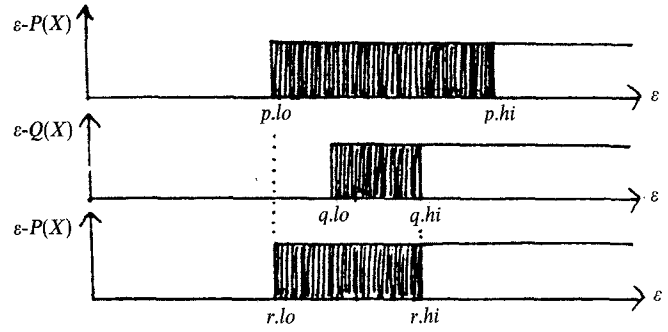

**Fig. 5.** Computing I_R from I_P and I_Q.

In general, given uncertainty intervals $p$ and $q$, we define the following two interval operations:

$$\mathtt{min}\{p, q\} = (\mathtt{min}\{p.lo, q.lo\},\ \mathtt{min}\{p.hi, q.hi\}),$$

$$\mathtt{max}\{p, q\} = (\mathtt{max}\{p.lo, q.lo\},\ \mathtt{max}\{p.hi, q.hi\}).$$

We use these operations as follows. Suppose we know that the predicate $\varepsilon$-$R(X)$ is equivalent to $\varepsilon$-$P(X) \vee \varepsilon$-$Q(X)$ for all $X$ and all $\varepsilon$. Then a procedure that computes the interval $\mathtt{min}\{\mathtt{I\_P}(X), \mathtt{I\_Q}(X)\}$ is a correct I-box for $R$. Similarly, if we know that $\varepsilon$-$R(X)$ is equivalent to $\varepsilon$-$P(X) \wedge \varepsilon$-$Q(X)$, then $\mathtt{max}\{\mathtt{I\_P}(X), \mathtt{I\_Q}(X)\}$ is a correct I-box for $R$.

The rules above are just two examples of how the results of epsilon boxes can be combined. Further examples can be found in our earlier paper [7].

*2.5. Basic Epsilon Predicates.* The algorithms we develop in this paper are built on top of a small set of geometric predicates for points in the plane—*Coincident*, *Collinear*, *Pos*, *Neg*, and *Between*—which we define below. We use the standard Euclidean metric to measure the size of perturbations for the epsilon versions of these predicates. The details of the implementations of these predicates are described in our previous paper [7].

*Coincidence.* The predicate *Coincident*$(p, q)$ merely tests whether the points $p$ and $q$ are the same. The derived predicate $\varepsilon$-*Coincident*$(p, q)$ is true if and only if points $p$ and $q$ lie within $\varepsilon$ of a common point, or, equivalently, if and only if $\|p, q\| \leq 2\varepsilon$. A geometric interpretation of this predicate requires that the $\varepsilon$-*disk* of $p$—that is, the closed disk of radius $\varepsilon$ centered on $p$—touch the $\varepsilon$-disk of $q$ (Figure 6).

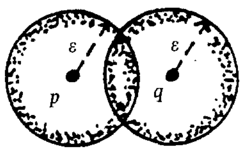

**Fig. 6.** $\varepsilon$-*Coincident*$(p, q)$.

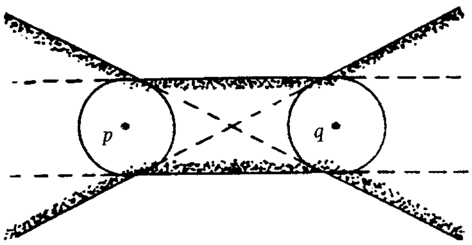

**Fig. 7.** The ε-butterfly of $p$ and $q$.

*Collinearity.* The predicate *Collinear*$(p, q, r)$ tests whether the points $p$, $q$, and $r$ lie on a common straight line, in any order. Therefore, ε-*Collinear*$(p, q, r)$ is true if and only if there exists a line $l$ that passes within ε of all three points. The predicates *Collinear* and ε-*Collinear* are obviously symmetric in their three arguments.

We can visualize the ε-*Collinear* predicate as follows. Let $P$ and $Q$ be the ε-disks of $p$ and $q$, respectively. Then the set of all lines passing through a point of $P$ and a point of $Q$ covers a butterfly-shaped region of the plane bounded by the two inner and two outer tangents of $P$ and $Q$. (If $P$ and $Q$ have a point in common, then this region degenerates to the entire plane.) We call this region the ε-***butterfly*** determined by $p$ and $q$ (Figure 7). The three points $p$, $q$, and $r$ are ε-collinear if and only if the ε-disk centered at $p$ intersects the ε-butterfly of $q$ and $r$ (Figure 8). Equivalently, the three points are ε-collinear if and only if one of the ε-disks intersects the ε-***stroke*** of the other two points, where the ε-stroke is defined as the convex hull of the two points' ε-disks.

*Orientation.* In exact geometry, a triangle $T = (p, q, r)$ whose vertices are not collinear can be further classified by its orientation, either *positive* (counterclockwise) or *negative* (clockwise). The orientation is the sign of the determinant

$$D(p, q, r) = \begin{vmatrix} 1 & p.x & p.y \\ 1 & q.x & q.y \\ 1 & r.x & r.y \end{vmatrix}.$$

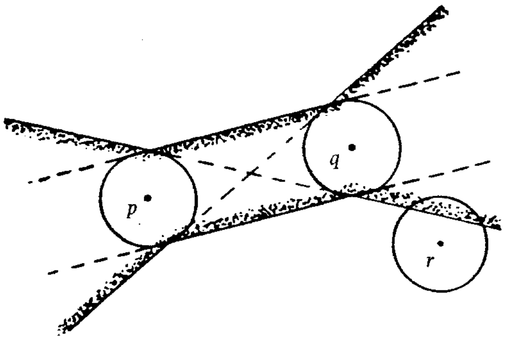

**Fig. 8.** ε-*Collinear*$(p, q, r)$.

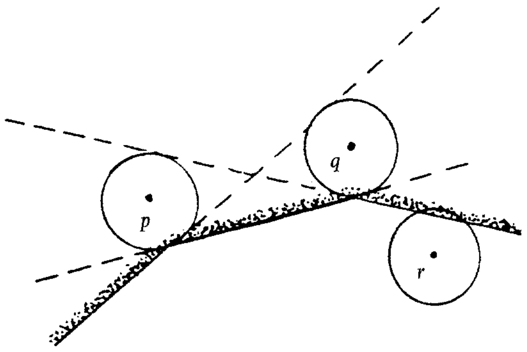

**Fig. 9.** $\varepsilon$-*Pos*$(p, q, r)$.

We define the predicates $Pos(p, q, r)$ and $Neg(p, q, r)$ as meaning $D(p, q, r) > 0$ and $D(p, q, r) < 0$, respectively. Note that $Pos(T)$ is not the same as `not` $Neg(T)$; in fact, `not` $Neg(T) \equiv Pos(T) \vee Collinear(T)$.

By definition then, $\varepsilon$-$Pos(p, q, r)$ means that it is possible to make $D(p, q, r) > 0$ by displacing the three points by at most $\varepsilon$ in suitable directions. In graphical terms, $\varepsilon$-$Pos(p, q, r)$ requires that the $\varepsilon$-disk centered at $p$ intersect the interior of the *left $\varepsilon$-basin* of $q$ and $r$, consisting of the $\varepsilon$-butterfly of $q$ and $r$ and all points to its left (when looking from $q$ toward $r$). See Figure 9.

From linear algebra we know that the determinant $D$ changes sign if we swap any two of the three points, and remains unchanged if the three points are permuted in a cyclic fashion. Thus,

$$\begin{aligned} &\varepsilon\text{-}Pos(p, q, r) \equiv \varepsilon\text{-}Pos(q, r, p) \equiv \varepsilon\text{-}Pos(r, p, q) \\ &\equiv \varepsilon\text{-}Neg(q, p, r) \equiv \varepsilon\text{-}Neg(r, q, p) \equiv \varepsilon\text{-}Neg(p, r, q). \end{aligned}$$

Note that $\varepsilon$-$Pos(T)$ is quite different from `not` $\varepsilon$-$Neg(T)$ and from $\varepsilon$-(`not` $Neg$)$(T)$. Instead, the following relationships hold:

$$\begin{aligned} \texttt{not}\ \varepsilon\text{-}Neg(T) &\equiv (-\varepsilon)\text{-}(\texttt{not}\ Neg)(T), \\ &\equiv (-\varepsilon)\text{-}(Pos \vee Collinear)(T). \end{aligned}$$

*Betweenness.* We say that a point $z$ lies *between* two other points $p$ and $q$ if $z$ lies on the closed segment $pq$. We denote this fact by $Between(z, pq)$. It is easy to see that $\varepsilon$-$Between(z, pq)$ is true if and only if the distance from $z$ to the segment $pq$ is at most $2\varepsilon$, or, equivalently, if and only if the $\varepsilon$-disk centered at $z$ intersects the $\varepsilon$-stroke of $p$ and $q$ (Figure 10).

Note that three points are $\varepsilon$-collinear if and only if at least one of the three points is $\varepsilon$-between the other two:

$$\varepsilon\text{-}Collinear(p, q, r) \equiv \varepsilon\text{-}Between(p, qr) \vee \varepsilon\text{-}Between(q, pr) \vee \varepsilon\text{-}Between(r, pq).$$

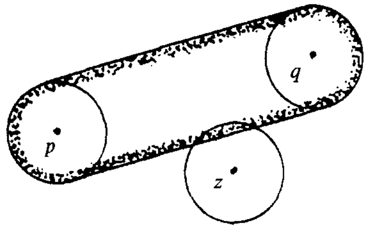

**Fig. 10.** $\varepsilon$-*Between*$(z, pq)$.

The following lemma, which we use later, captures another useful relationship between the *Between* and *Collinear* predicates:

LEMMA 4. *Let $a$, $b$, and $x$ be three points such that $x$ is inside a closed rectangle with diagonal $ab$. Then $\varepsilon$-Between$(x, ab) \Leftrightarrow \varepsilon$-Collinear$(x, a, b)$, for all $\varepsilon$.*

PROOF. For $\varepsilon < 0$, the predicates $\varepsilon$-*Between*$(x, ab)$ and $\varepsilon$-*Collinear*$(x, a, b)$ are both false. For $\varepsilon \geq 0$, the smallest perturbation needed to make the points $a$, $x$, and $b$ collinear is half the smallest altitude of the triangle $axb$. Since $x$ is inside a right triangle with hypotenuse $ab$, the angle at $x$ is 90° or more, and therefore the triangle's shortest altitude is the perpendicular distance from $x$ to segment $ab$. □

## 3. Strongly Convex Polygons

*3.1. Definitions.* For this paper we define a *polygon* $P$ to be a sequence of vertices $(p_0, \ldots, p_{n-1})$. For convenience, we let $p_i$ stand for the vertex $p_{i \bmod n}$ for all integers $i$. The *edges* of $P$ are the closed segments $p_i p_{i+1}$. The polygon is *simple* if two edges $p_i p_{i+1}$ and $p_j p_{j+1}$ intersect only when $j = i - 1$, $j = i$, or $j = i + 1$ (modulo $n$).

The *external angle* of a polygon $P = (p_0, \ldots, p_{n-1})$ at vertex $p_i$ is the angle from the vector $p_i - p_{i-1}$ to the vector $p_{i+1} - p_i$, measured counterclockwise and reduced to the interval $(-\pi, \pi)$. The external angle is undefined if the two vectors point in opposite directions, or if one of them is zero. The *degree* $\partial P$ of $P$ is the sum of its external angles, divided by $2\pi$. It is well known that the degree of a polygon (when defined) is an integer, and that the degree of a simple polygon is always defined and is either $+1$ or $-1$ [6], [8].

A polygon $P$ is *convex* if all vertices $p_i$, $p_j$ with $i < j < i + n$ are distinct, and all triples $p_i$, $p_j$, $p_k$ with $i < j < k < i + n$ are (strictly) positive. Note that by this definition a single point, as well as a pair of noncoincident points, qualifies as a convex polygon. Note also that every subpolygon of a convex polygon is convex.

Convex polygons have several useful properties. We say that a periodic sequence of real numbers is *unimodal* if it has at most one ascending run and one descending run in one period (allowing for repeated elements), and that a polygon is *d-unimodal* for some direction $d$ if the projection of its vertices on a line parallel to $d$ forms a unimodal sequence. We say that a polygon $P$ with three or more vertices is *left-turning* if all its exterior angles are defined and positive, that is, if every triple of consecutive vertices $p_{i-1}p_ip_{i+1}$ is positively oriented. Then the following results are well known [8]:

LEMMA 5. *For any polygon $P = (p_0, \dots, p_{n-1})$, with $n \geq 3$, and any direction $d$, the following statements are equivalent*:

1. *$P$ is convex.*
2. *$P$ is left-turning and simple.*
3. *$P$ is left-turning and has degree $+1$.*
4. *$P$ is left-turning and $d$-unimodal.*

*3.2. Strong Convexity.* According to our definition of epsilon predicates, we say that a polygon is $(-\varepsilon)$-convex, for some $\varepsilon \geq 0$, if it remains convex under any perturbation to its vertices of $\varepsilon$ or less. Formally, when measuring convexity, we define the "distance" between two polygons as the maximum distance between a vertex of one polygon and the corresponding vertex of the other. Two polygons with a different number of vertices are defined to be infinitely far apart. We informally use the term *strongly convex* to mean $(-\varepsilon)$-convex for some $\varepsilon > 0$ implied by the context.

Note that a $(-\varepsilon)$-convex polygon automatically satisfies a kind of "minimum feature separation" condition, similar to the explicit conditions required by other robust algorithms in the literature [11], [14], [15], [22]. For example, in a $(-\varepsilon)$-convex polygon, no two vertices may be $\varepsilon$-coincident, and, as we shall see, every vertex must lie more than $2\varepsilon$ away from any diagonal.

The following theorems characterize strongly convex polygons:

THEOREM 6. *A polygon $P = (p_0, \dots, p_{n-1})$ is $(-\varepsilon)$-convex for some $\varepsilon \geq 0$ if and only if no two vertices are $\varepsilon$-coincident and all triples $p_ip_jp_k$ with $i < j < k < i + n$ are $(-\varepsilon)$-positive.*

PROOF. The theorem follows directly from the definition of convex polygon and Lemma 2. □

THEOREM 7. *A polygon $P = (p_0, \dots, p_{n-1})$, with $n \geq 3$, is $(-\varepsilon)$-convex from some $\varepsilon \geq 0$ if and only if $P$ is convex and every consecutive triple of vertices $p_{i-1}p_ip_{i+1}$ is $(-\varepsilon)$-positive.*

PROOF. The forward implication follows immediately from the definition of convexity and from Lemma 2. As for the converse, suppose $P$ is convex and all consecutive triples of vertices are $(-\varepsilon)$-positive. We must prove that any perturbed version $P' = (p'_0, \dots, p'_{n-1})$ of $P$, such that $\|p'_i, p_i\| \leq \varepsilon$ for all $i$, is also convex. Note that $P'$ is at least left-turning, because every consecutive triple $p'_{i-1}p'_ip'_{i+1}$ is still positive. Therefore, by Lemma 5, we need only prove that $P'$ has degree $+1$.

Consider the family of polygons $P^\lambda = (p_0^\lambda, \dots, p_{n-1}^\lambda)$, where $p_i^\lambda = (1 - \lambda)p_i + \lambda p'_i$, for all real $\lambda$ in the interval [0_1]. Note that $P^0 = P$, $P^1 = P'$, and that the position of each vertex $p_i^\lambda$ is a continuous function of $\lambda$. Note also that $\|p_i^\lambda, p_i\| \leq \varepsilon$ for all

$i$. Since all consecutive triples $p_{i-1}p_ip_{i+1}$ of $P$ are $(-\varepsilon)$-positive, it follows that all consecutive triples $p^\lambda_{i-1}p^\lambda_i p^\lambda_{i+1}$ are positive for any $\lambda$ in [0_1].

The external angle of $P^\lambda$ at each vertex $p^\lambda_i$ is therefore a well-defined, continuous function of $\lambda$ for all $\lambda$ in [0_1]. The same must therefore be true of the total degree $\partial(P^\lambda)$. However, we know that $\partial(P^0)$ is $+1$ since $P^0 = P$ is a convex polygon, and also that $\partial(P^\lambda)$ is integral when defined; hence, $\partial(P^1) = \partial(P')$ is also $+1$. □

THEOREM 8. *A polygon $P = (p_0, \ldots, p_{n-1})$, with $n \geq 3$, is $(-\varepsilon)$-convex for some $\varepsilon \geq 0$ if and only if every consecutive triple of vertices $p_{i-1}p_ip_{i+1}$ is $(-\varepsilon)$-positive, and any of the following conditions hold*:

1. *P is simple.*
2. *P has degree $+1$.*
3. *P is unimodal in some direction d.*

PROOF. The forward implication follows immediately from Lemmas 2 and 5. The converse follows from Lemma 5 and Theorem 7. □

Theorem 8 gives us an $O(n)$-time algorithm for testing whether a polygon is convex, and if so, for measuring how convex it is. First, we test (using exact comparisons in $x$) whether the polygon is $x$-unimodal. If not, we return the uncertainty interval $(0, +\infty)$. Otherwise, we compute $d = \mathtt{max}_i\mathtt{I\_Pos}(p_{i-1}, p_i, p_{i+1})$. If $d.hi < 0$, then, by Theorem 8, the polygon is $d$-convex, and we can return $d$. Otherwise, we must return the interval $(d.lo, +\infty)$.

The algorithm is $\lambda$-accurate, provided the I_Pos box it uses is $\lambda$-accurate and the polygon is $(-\lambda)$-convex. Note that the algorithm gives no quantitative information if the polygon is not convex or is just barely convex, since, as Figure 11 shows, all consecutive triples may be nearly convex while the polygon as a whole is arbitrarily far from being convex. The best $O(\lambda)$-accurate algorithm we know for measuring the nonconvexity of a nonconvex polygon takes $O(n^3)$ time [21].

In the following sections we also require the following characterization of strongly convex polygons, which is similar to Theorem 7 but with a slightly weaker condition:

LEMMA 9. *A polygon $P = (p_0, \ldots, p_{n-1})$ is $(-\varepsilon)$-convex for some $\varepsilon \geq 0$ if and only if P is convex and, if $n > 1$, no vertex $p_i$ of P is $\varepsilon$-between its neighbors $p_{i-1}$ and $p_{i+1}$.*

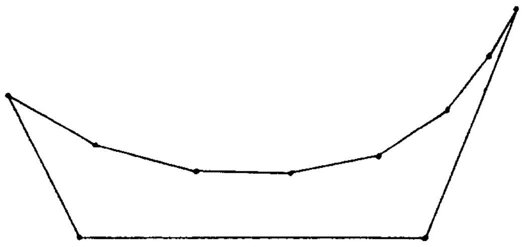

**Fig. 11.** An extremely nonconvex polygon made of nearly convex triples.

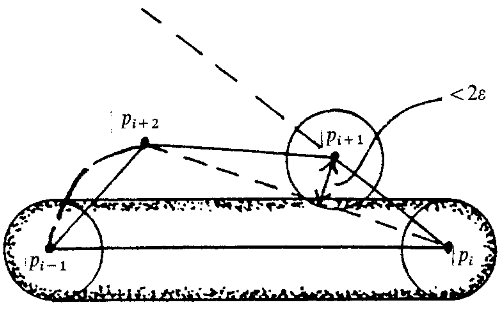

**Fig. 12.** Construction for the proof of Lemma 9.

PROOF. For $n = 1$ or $n = 2$, the lemma follows trivially from the definition of a convex polygon. For $n \geq 3$, the forward implication follows immediately from Theorem 7. As for the converse, suppose $P$ is convex with no vertex $p_i$ $\varepsilon$-between its two neighbors; we must prove that $P$ is $(-\varepsilon)$-convex. Because of Theorem 7, it suffices to prove that every consecutive triple of vertices is $(-\varepsilon)$-positive.

Suppose that the opposite were true, namely, that some triple $p_{i-1}p_ip_{i+1}$ is not $(-\varepsilon)$-positive. Since $P$ is convex, the triple is at least positive; it follows that the triple is actually $\varepsilon$-collinear. Therefore, at least one vertex $p_{i-1}$, $p_i$, or $p_{i+1}$ must lie $\varepsilon$-between the other two, and we know by hypothesis it is not $p_i$. Without loss of generality, we assume that $p_{i+1}$ lies $\varepsilon$-between $p_{i-1}$ and $p_i$.

Observe that polygon $P$ must have at least four vertices, since otherwise $p_{i+1}$ would lie $\varepsilon$-between its two neighbors. By convexity, the next vertex $p_{i+2}$ must lie strictly inside the angle determined by $p_{i-1}$, $p_i$, and $p_{i+1}$ (Figure 12). Now observe that the shortest segment from $p_{i+1}$ to the segment $p_{i-1}p_i$ must intersect segment $p_{i+2}p_i$, since the quadrilateral $(p_{i-1}, p_i, p_{i+1}, p_{i+2})$ is convex. Thus, vertex $p_{i+1}$ must lie at least as close to segment $p_ip_{i+2}$ as it does to segment $p_{i-1}p_i$. However, by hypothesis $p_{i+1}$ lies $\varepsilon$-between the vertices $p_{i-1}$ and $p_i$, so it must also lie $\varepsilon$-between the vertices $p_i$ and $p_{i+2}$, a contradiction.

We conclude that all triples of $P$ are $(-\varepsilon)$-positive, and therefore $P$ is $(-\varepsilon)$-convex. □

Here is another useful property of strongly convex polygons:

THEOREM 10. *Let $\varepsilon$ and $\delta$ be positive real numbers. For any $(-\varepsilon)$-convex polygon $P$ and any two points $u$ and $v$ in the plane, there are at most $2\lceil 2\delta/\varepsilon \rceil$ vertices of $P$ that are $\delta$-between the points $u$ and $v$.*

PROOF. Consider the infinite strip of width $4\delta$ centered on the line $uv$, together with a set of lines parallel to $uv$ lying inside the strip and placed so that every point of the strip is at most $\varepsilon$ away from one of these lines (Figure 13).

We can always achieve this covering with at most $\lceil 2\delta/\varepsilon \rceil$ lines. Because the polygon $P$ is $(-\varepsilon)$-convex, each of these lines can intersect at most two of the $\varepsilon$-disks centered at the vertices of $P$. On the other hand, if a vertex of $P$ is inside the strip, then its $\varepsilon$-disk must intersect one of these lines. Therefore, the total number of vertices of $P$ that lie inside the strip is at most twice the number of

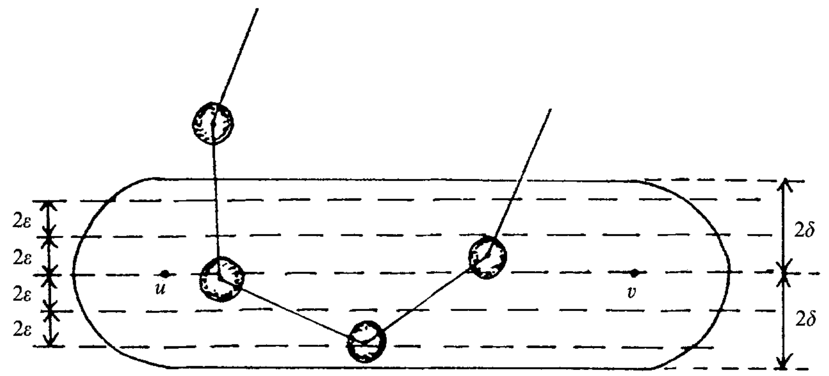

**Fig. 13.** Construction for the proof of Theorem 10.

lines. Since any point of the plane that is $\delta$-between points $u$ and $v$ must lie inside this strip, we conclude that at most $2\lceil 2\delta/\varepsilon \rceil$ vertices of $P$ are $\delta$-between the points $u$ and $v$. □

As an aside, we note that the number of vertices of any $(-\varepsilon)$-convex polygon contained in a circle of radius $R$ is bounded. A simple geometric argument provides an upper bound of $2\pi/\arccos(1 - 2\varepsilon/R)$ vertices, which reduces to $\pi\sqrt{R/\varepsilon} + O((\varepsilon/R)^{3/2})$ for small $\varepsilon/R$. Thus, for example, if $\varepsilon/R = 10^{-6}$, a typical uncertainty for floating point operations, then any such polygon is guaranteed to have fewer than 3200 vertices.

## 4. A Point Inclusion Algorithm.

As an application of these theorems, let us consider the problem of testing whether a given point $z$ lies inside an $n$-sided convex polygon $P$. (We consider point $z$ to lie "inside" $P$ if it lies either on the interior or on the boundary of $P$.)

A classical $O(\log n)$-time algorithm for solving this problem begins by locating the point $z$ in the angle between two consecutive diagonals $p_0 p_k$ and $p_0 p_{k+1}$, using binary search, and then testing $z$ against the line $p_k p_{k+1}$ [20]. Note that this algorithm can be expressed using just the *Pos* orientation test.

Unfortunately, this simple algorithm no longer works if the exact *Pos* test is replaced by an approximate one. Figure 14 illustrates the problem: in both cases, the point $z$ lies approximately within the angle $p_{k+1} p_0 p_k$ and on the positive side of edge $p_k p_{k+1}$, but in one case it lies inside the polygon and in the other case it lies well outside.

The key idea in the classical point-location algorithm is that if $z$ is found to lie on the positive side of a diagonal $p_0 p_k$, then the subpolygon $(p_0, p_1, \ldots, p_k)$ can be eliminated from further consideration. We can still use this idea—provided that the polygon is sufficiently convex—thanks to the following lemma:

LEMMA 11. *Let $P = (p_0, \ldots, p_{n-1})$ be a $(-\varepsilon)$-convex polygon for some $\varepsilon \geq 0$. Then the vertices $p_j, \ldots, p_{k-j}$ are more than $(2j - 1)\varepsilon$ away from the left $\varepsilon$-basin of $p_0 p_k$ for any $j$, $k$ such that $0 < j < k < n$.*

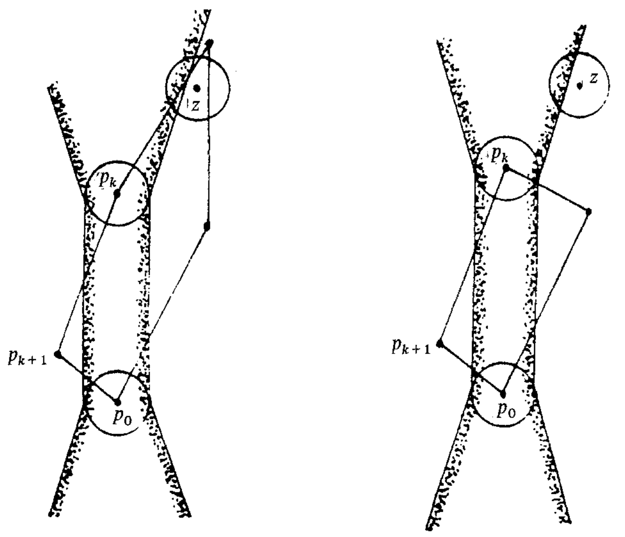

**Fig. 14.** Two indistinguishable cases.

PROOF. The case $j = 1$ follows from the definition of $(-\varepsilon)$-convexity.

We prove the remaining cases by induction on $j$. Suppose we have already proved the lemma for the case $j$, and we want to prove it for the case $j + 1$, that is, that the vertices $p_{j+1}, \ldots, p_{k-j-1}$ lie more than $(2j + 1)\varepsilon$ away from the $\varepsilon$-butterfly of $p_0 p_k$.

We introduce the following notation. Consider the two $\varepsilon$-disks centered at points $p$ and $q$. We denote by $p^+q^+$ the oriented line that is tangent to those two disks in the given order and that leaves the two disks to its left. Similarly, we let $p^+q^-$ be the tangent to the disks that leaves $p$ to its left and $q$ to its right, and so on. We also denote by $L(m)$ and $R(m)$ the open left and right half-planes of an oriented line $m$.

Assume that the polygon's vertices are numbered counterclockwise. By the definition of $(-\varepsilon)$-convexity, the disks of $p_{j+1}$ and $p_{k-j-1}$ must lie in the intersection $X$ of the half-planes $R(r)$, $L(s)$, $R(t)$, where $r = p_k^- p_{k-j}^+$, $s = p_0^+ p_j^-$, and $t = p_j^+ p_{k-j}^+$ (Figure 15). Let $Y$ be the (open) set of points that are more than $(2j - 2)\varepsilon$ away from the left $\varepsilon$-basin of $p_0 p_k$. We show that $X$ lies inside $Y$ and at least $2\varepsilon$ from its boundary.

Without loss of generality, we can assume that the line $s$ coincides with the $x$-axis. Let $L$ be the set of all oriented lines that pass from the $\varepsilon$-disk of $p_0$ to the $\varepsilon$-disk of $p_k$. From the definition of $(-\varepsilon)$-convexity we know that the vertex $p_k$ must lie at least $3\varepsilon$ above the $x$-axis, so all lines of $L$ are directed upward. Since $Y$ is the intersection of the right half-planes of lines that are parallel to lines of $L$, the boundary of $Y$ is a single $y$-monotone chain. Therefore, since the $\varepsilon$-disk of $p_j$ lies inside $Y$, the entire semi-infinite $\varepsilon$-stroke starting at $p_j$ and extending horizontally and to the right also lies inside $Y$. By an entirely symmetric argument, the semi-infinite $\varepsilon$-stroke extending from $p_{k-j}$ along the line $r$ must also lie in $Y$. Finally,

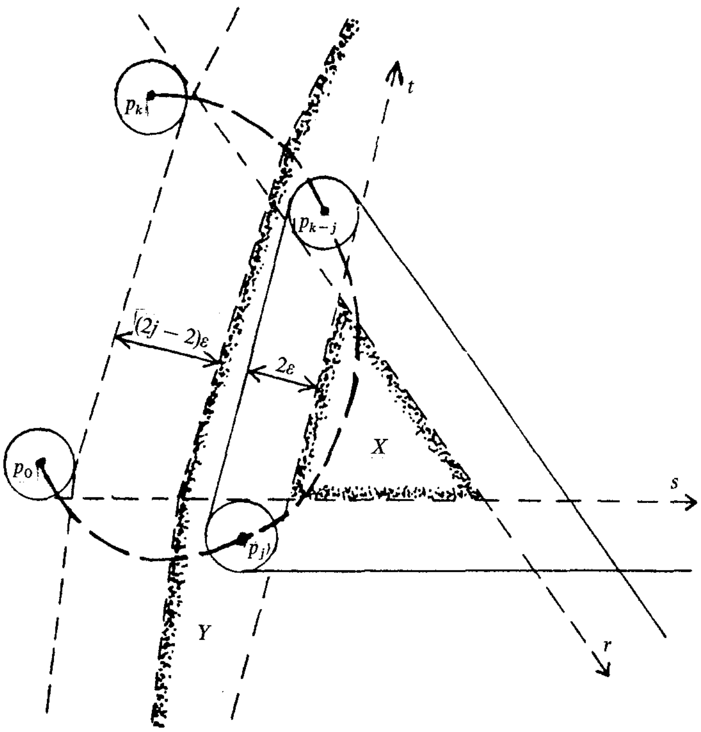

**Fig. 15.** Construction for the proof of Lemma 11.

since the $\varepsilon$-disks of $p_j$ and $p_{k-j}$ lie inside $Y$, and $Y$ is convex, the entire $\varepsilon$-stroke of $p_jp_{k-j}$ also lies inside $Y$. The union of these three strokes forms a border around the region $X$ that is at least $2\varepsilon$ thick and guarantees that every point of $X$ is more than $2\varepsilon$ away from the boundary of $Y$.

Since the $\varepsilon$-disks of $p_{j+1}, \ldots, p_{k-j-1}$ all lie entirely within $X$, we conclude that each of these points lies more than $\varepsilon + 2\varepsilon + (2j-2)\varepsilon = (2j+1)\varepsilon$ away from the $\varepsilon$-butterfly of $p_0p_k$. □

Here, then, is an accurate $O(\log n)$-time T-box for the *InStronglyConvex* primitive. In addition to the `T_Pos` box already defined, the algorithm uses the accurate epsilon box `T_InConvex` $(z, D)$, which tests in $O(n)$ time whether the point $z$ lies inside the convex polygon $D$, and which is described in our previous paper [7].

ALGORITHM 1. Given a point $z$ and a $(-\varepsilon)$-convex polygon $P = (p_0, \ldots, p_{n-1})$, for sufficiently large $\varepsilon$, determine whether $z$ is $\varepsilon$-inside $P$.

```
1. Initialize $D \leftarrow P$.
2. While $D$ has 12 or more vertices, do:
   a. Let the vertices of $D$ be $(d_0, d_1, \ldots, d_{m-1})$. Let $k = \lfloor m/2 \rfloor$.
   b. Let $t = $ T_Pos$(z, d_0, d_k)$.
```

c. If $t$ is `true` or `unknown`, delete the vertices $d_3, \ldots, d_{k-3}$ from $D$.
d. If $t$ is `false` or `unknown`, delete the vertices $d_{k+3}, \ldots, d_{m-3}$ from $D$.

3. Return `T_InConvex`$(z, D)$.

Clearly, the algorithm runs in $O(\log n)$ time. Note also that the vertices removed from $P$ comprise at most two chains of consecutive vertices of $P$, namely $(p_3, \ldots, p_{i-3})$ and $(p_{j+3}, \ldots, p_{n-3})$, for some $i, j$. Thus, polygon $D$ can be represented by the original array of vertices, along with the two indices $i$ and $j$.

The algorithm requires that $\varepsilon$ be greater than or equal to the maximum uncertainty $\lambda$ of the primitive epsilon boxes `T_Pos` and `T_InConvex`. Assuming this condition is satisfied, the correctness and accuracy of the algorithm are implied by the following invariants, which hold at the beginning of step 2a:

$$\begin{aligned} \mathit{Inside}(z, P) &\Leftrightarrow \mathit{Inside}(z, D), \\ \varepsilon\text{-}\mathit{Boundary}(z, P) &\Leftrightarrow \varepsilon\text{-}\mathit{Boundary}(z, D), \end{aligned}$$

where *Boundary*$(z, P)$ means that point $z$ lies on the boundary of $P$. (Note that the predicate $\varepsilon$-*Boundary*$(z, P)$ is equivalent to $\bigvee_i$ $\varepsilon$-*Between*$(z, p_i p_{i+1})$.) These invariants are implied by the following lemma:

Lemma 12. *Let $D = (d_0, \ldots, d_{m-1})$ be a $(-\varepsilon)$-convex polygon for some $\varepsilon > 0$, let $z$ be a point such that $\varepsilon$-Pos$(z, d_0, d_k)$ for some $5 \leq k \leq m - 5$, and let $D'$ be the subpolygon obtained by deleting the vertices $d_3, \ldots, d_{k-3}$ from $D$. Then*

$$\begin{aligned} \mathit{Inside}(z, D) &\Leftrightarrow \mathit{Inside}(z, D'), \\ \varepsilon\text{-}\mathit{Boundary}(z, D) &\Leftrightarrow \varepsilon\text{-}\mathit{Boundary}(z, D'). \end{aligned}$$

Proof. By hypothesis, the point $z$ must lie less than $\varepsilon$ away from the left $\varepsilon$-basin $X$ of $d_0 d_k$. By Lemma 11, the vertices $d_2, \ldots, d_{k-2}$ are more than $3\varepsilon$ away from $X$. By convexity, the same is true for the entire subpolygon $D''$ of $D$ determined by these vertices. Therefore, the point $z$ must lie more than $2\varepsilon$ away from $D''$. Since $D = D' \cup D''$, it follows that *Inside*$(z, D) \Leftrightarrow$ *Inside*$(z, D')$.

Furthermore, note that the only differences between the boundaries of $D$ and $D'$ are the edges of $D''$. Since $\varepsilon$-*Boundary*$(z, D)$ means that $z$ lies within $2\varepsilon$ of the boundary of $D$, and since all edges of $D''$ lie more than $2\varepsilon$ from the boundary of $D$, we conclude that $\varepsilon$-*Boundary*$(z, D) \Leftrightarrow \varepsilon$-*Boundary*$(z, D')$. □

Note that it is possible to design a simpler point-location algorithm by first using exact $x$-comparisons to locate the point in the vertical slab between consecutive vertices of the polygon, and by then using approximate orientation tests only to test the point against the two edges of the polygon that enter the slab. As this example shows, the use of exact comparisons between real numbers often leads to simpler algorithms for solving problems in the plane. One reason for this is that $k$-dimensional geometric problems can often be solved by reducing them to $(k-1)$-dimensional subproblems, and one-dimensional subproblems can often be solved exactly using exact comparisons. Indeed, Fortune [3] and Milenkovic [15] have chosen to take this approach in their algorithms.

However, we expect exact comparisons to be less useful for solving problems in three or more dimensions. For instance, the naive three-dimensional generalization of the slab-based algorithm above would not work, because it would require locating the point on a two-dimensional subdivision (the projection of the polyhedron on some plane), which cannot be performed exactly, even assuming exact $x$- and $y$-comparisons; and also because there is no three-dimensional analog of Lemma 4, which is used implicitly in the slab-based algorithm to guarantee that the point is $\varepsilon$ away from $P$ if and only if it is $\varepsilon$ away from the tested edges. In view of these considerations, we have found it worthwhile to develop a point-location algorithm that does not use exact comparisons, in the hope that it may be easier to generalize to higher dimensions.

## 5. Strongly Convex Hulls.

Given a set of points $S$, we say that $P \subseteq S$ is a $\delta$-hull for $S$ if every point of $S$ is $\delta$-inside $P$. This definition agrees with our general notion of epsilon predicates if we define the distance between two polygons to be the maximum distance between a vertex of one polygon and the nearest point on the boundary of the other. Note that this metric is different from the one used for measuring convexity, since it does not require vertices to be matched one-to-one.

### 5.1. *Properties of Strongly Convex Hulls.*

The $\varepsilon$-convex $\delta$-hull of a set $S$ is not necessarily unique for $\delta > 0$. However, we can at least prove that any two such hulls are not very different from one another. The following theorems make this notion more precise:

LEMMA 13. *For any four points $u$, $v$, $u'$, $v'$ such that $\|u, u'\| \le \varepsilon$ and $\|v, v'\| \le \varepsilon$, and for any $\alpha \in [0\_1]$, we have*

$$\|(1-\alpha)u + \alpha v, (1-\alpha)u' + \alpha v'\| \le \varepsilon. \tag{3}$$

PROOF. Observe that the distance (3) is the length of the vector $(1-\alpha)(u-u') + \alpha(v-v')$. We know that the vectors $u-u'$ and $v-v'$ lie inside the $\varepsilon$-disk centered at the origin of $\Re^2$. The result follows from the convexity of the $\varepsilon$-disk. □

THEOREM 14. *If $G$ and $H$ are convex $\delta$-hulls of $S$ with $\delta \ge 0$, then every point on the boundary of $G$ is $\delta$-inside $H$, and vice versa.*

PROOF. By definition, every vertex of $G$ is $\delta$-inside $H$, that is, at most $2\delta$ away from the boundary of $H$. Let $g'$ and $g''$ be two consecutive vertices of $G$. Let $h'$ and $h''$ be the points (not necessarily vertices) of $H$ that are closest to $g'$ and $g''$, respectively. Let $g$ be any point on the edge $g'g''$. By Lemma 13, there is a point $h$ on the segment $h'h''$ that is at most $2\delta$ away from $g$. The point $h$ is inside the polygon $H$ by convexity, and therefore the point $g$ is $\delta$-inside $H$. By the same argument, every point on the boundary of $H$ is also $\delta$-inside $G$. □

Corollary 15. *If $G$ and $H$ are convex $\delta$-hulls of $S$ with $\delta \geq 0$, then every point on the boundary of $G$ is $\delta$-boundary of $H$, and vice versa.*

The following theorem shows that the number of vertices of any $(-\varepsilon)$-convex $(\delta)$-hull of $S$ is within a constant factor of the number of vertices of the smallest such hull.

Theorem 16. *If $G$ and $H$ are $(-\varepsilon)$-convex $\delta$-hulls of $S$ with $\varepsilon$, $\delta > 0$, then $|G| \leq 2\lceil 2\delta/\varepsilon \rceil |H|$.*

Proof. By Corollary 15, every vertex of $G$ is $\delta$-between two consecutive vertices of $H$. By Theorem 10, no more than $M = 2\lceil 2\delta/\varepsilon \rceil$ of these vertices can be $\delta$-between any two given points. The theorem then follows trivially. □

*5.2. Existence of Strongly Convex Hulls.* It is by no means obvious that a $(-\varepsilon)$-convex $\delta$-hull always exists for an arbitrary set of points $S$ and an arbitrary (positive) $\varepsilon$ and $\delta$. In fact, when $\delta < \varepsilon$, there may not exist any such hull. As a counterexample, consider a set of four points arranged on the vertices of a square so that each point is just barely $\varepsilon$-between its two neighbors (Figure 16). In order to ensure $(-\varepsilon)$-convexity, only two of the four points can be chosen for the hull, leaving the other two points arbitrarily close to lying $2\varepsilon$ away.

On the other hand, we prove here that a $(-\varepsilon)$-convex $\delta$-hull always exists whenever $\delta \geq 2\varepsilon$. (The existence question is still open for $\varepsilon \leq \delta < 2\varepsilon$.) We begin with a lemma:

Lemma 17. *If $P = (p_0, \ldots, p_{n-1})$ is a convex polygon, then there exists a subpolygon $H$ of $P$ that is $(-\varepsilon)$-convex, and that is a $2\varepsilon$-hull for $P$.*

Proof. For the purposes of this proof, we extend $P$ with an extra vertex $p_n$ coincident with $p_0$. We say that a vertex $p_i$ of a polygon is *$\varepsilon$-flat* if $p_i$ is $\varepsilon$-between its two neighbors $p_{i-1}$ and $p_{i+1}$. We say that a vertex $p_j$ *covers* a vertex $p_k$ if $j \leq k < j + n$ and, for every $i$ such that $j < i < k$, the point $p_i$ is $\varepsilon$-between the points $p_j$ and $p_k$.

We denote by $c(k)$, for $0 \leq k \leq n$, the smallest $j$ in $\{0, \ldots, k-1\}$ such that $p_j$ covers $p_k$. Note that $c(0) = 0$ and $c(k) < k$ for all $k > 0$. Note also that $c$ is

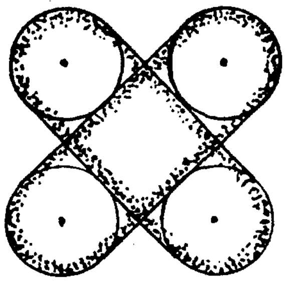

**Fig. 16.** Counterexample for the existence of a $(-\varepsilon)$-convex $\delta$-hull for $\delta < \varepsilon$.

monotonically nondecreasing: $k \le l$ implies $c(k) \le c(l)$. Furthermore, every vertex between $p_{c(k)}$ and $p_k$, inclusive, covers $p_k$.

We now recursively define a second function $s(k)$, for $0 \le k \le n$, as follows: $s(k)$ is 0 if $c(k)$ is 0; otherwise, $s(k)$ is the smallest $j$ such that $p_j$ covers $p_k$, and $p_j$ is not $\varepsilon$-between vertices $p_{s(j)}$ and $p_k$. Note that $s(k) \ge c(k)$, and $s(k) < k$ except when $k = 0$.

Assuming that $s(k)$ indeed exists for all $k$, then the function $s$ defines for each vertex $p_k$ an implicit chain of vertices $H(k)$ that begins with $p_0$, ends with $p_k$, and is such that for every vertex $p_j$ except $p_0$ the previous vertex in the chain is $p_{s(j)}$. By construction, the chain $H(k)$ has the property that every interior vertex $p_j$ covers the next vertex in the chain and is not $\varepsilon$-between the two vertices adjacent to it in the chain. Therefore, the chain $H(n)$ defines a polygon $H$ that is an $\varepsilon$-hull for $P$, and that has no $\varepsilon$-flat vertices, with the possible exception of $p_0$. If vertex $p_0$ is also not $\varepsilon$-flat, then $H$ is $(-\varepsilon)$-convex, and we are done. Otherwise, by removing $p_0$ from $H$, we obtain a polygon that is a $(-\varepsilon)$-convex $2\varepsilon$-hull for $P$, and the lemma is still true.

It remains to prove that the function $s$ is well defined for all $k$. The proof is by induction: $s(0)$ is 0 by definition, and we prove that $s(k)$ exists by assuming that $s(0), \ldots, s(k-1)$ exist. It suffices to show that there exists a "good" index $i$ in $\{c(k), \ldots, k-1\}$ such that $p_i$ is not $\varepsilon$-between $p_{s(i)}$ and $p_k$.

Let $j = c(k)$. If $j = 0$, then $s(k) = 0$ by definition, and we are done. Otherwise, the definition of $c(k)$ implies that $p_{j-1}$ does not cover $p_k$. In other words, there is some vertex in $\{p_j, \ldots, p_{k-1}\}$ that is not $\varepsilon$-between the vertices $p_{j-1}$ and $p_k$. Let $p_m$ be the first such vertex, and let $i$ be $s(m)$. We have two cases (Figure 17):

If $i < j$, then $m$ is a good index, since segment $p_{j-1}p_k$ is the diagonal of a convex quadrilateral $(p_i, p_{j-1}, p_m, p_k)$, and therefore the distance from $p_m$ to $p_ip_k$ is greater

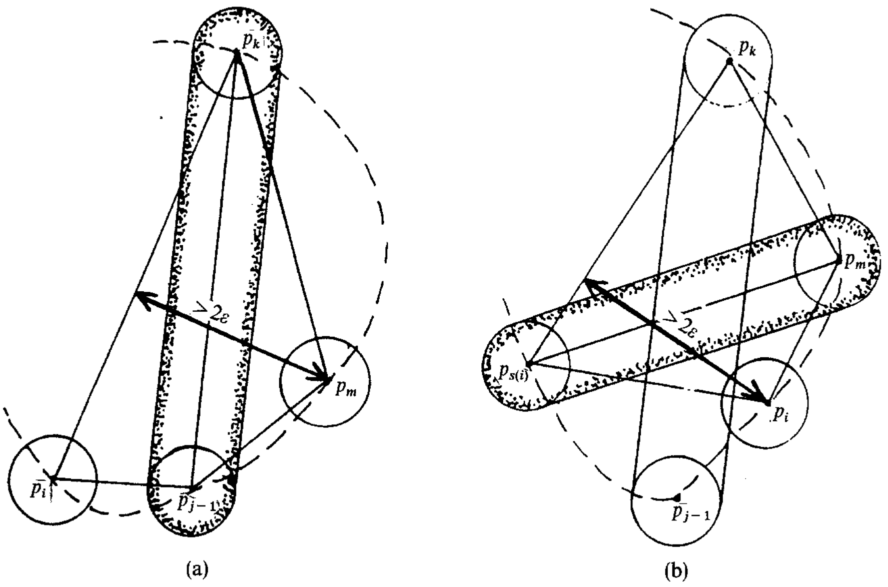

**Fig. 17.** Construction for the proof of Lemma 17.

than the distance from $p_m$ to $p_{j-1}p_k$, which we already know is greater than $2\varepsilon$ (Figure 17(a)).

If $i \geq j$, then $i$ is a good index, since segment $p_{s(i)}p_m$ is the diagonal of a convex quadrilateral $(p_{s(i)}, p_i, p_m, p_k)$, and therefore the distance from $p_i$ to $p_{s(i)}p_k$ is greater than the distance from $p_i$ to $p_{s(i)}p_m$, which we already know is greater than $2\varepsilon$ (Figure 17(b)). □

THEOREM 18. *For any set of points $S$, there is a subset $H$ of $S$ such that the points of $H$ are the vertices of a $(-\varepsilon)$-convex $2\varepsilon$-hull of $S$.*

PROOF. Let $P$ be the ordinary convex hull of $S$. By Lemma 17, there is a subpolygon $H$ of $P$ that is $(-\varepsilon)$-convex and is such that every point of $P$ is $2\varepsilon$-inside $H$. However, since every point of $S$ is inside $P$, every point of $S$ must also lie $2\varepsilon$-inside $H$, so $H$ is a $(-\varepsilon)$-convex $2\varepsilon$-hull for $S$. □

## 5.3. *Computing Strongly Convex Hulls*

ALGORITHM 2. Given a set $S$ of $n$ points in the plane and an $\varepsilon \geq 0$, return a $(-\varepsilon)$-convex polygon $H$, and an uncertainty interval $d$, such that $H$ is a $d$-hull for $S$. The algorithm guarantees that $d.hi \leq 6\varepsilon + 7\lambda$, where $\lambda$ is the maximum uncertainty of the primitive epsilon boxes called by the algorithm.

1. Find the $x$- and $y$-extremal points of $S$. Call then $t_0$, $t_1$, $t_2$, $t_3$, in counterclockwise order.
2. For each consecutive pair $t_q t_{q+1}$, do:
   a. Let $S_q$ be the set of all points $s \in S$ such that $\text{E\_Pos}(-\varepsilon, t_q, s, t_{q+1}) =$ `true`.
   b. Make $S_q$ into an $xy$-monotone chain $C_q$ by sorting the points in $x$ and $y$ and removing all points that are $xy$-dominated by other points in quadrant $q$. (For example, in the upper right quadrant, a point is $xy$-dominated if its $x$- and $y$-coordinates are both less than those of some other point.) Let $(c_0, \ldots, c_{m-1})$ be the points of this chain $C_q$ from $t_q$ to $t_{q+1}$, inclusive.
   c. Build a graph $G_q = (N, A)$ as follows:

$$N = \{(c_i c_j) : i < j\},$$
$$A = \{((c_i c_j), (c_j c_k)) : i < j < k \wedge \text{E\_Pos}(-\varepsilon, c_i, c_j, c_k) = \texttt{true}\}.$$

   d. Compute for each node $(c_i c_j)$ in $N$ a penalty $f(c_i c_j)$, defined as the interval $\texttt{max}\{d_{irj} : i < r < j\}$, where $d_{irj} = \text{I\_Neg}(c_i, c_r, c_j)$.
   e. Define the penalty $f(P)$ of any directed path $P$ in this graph to be the maximum of the penalties for the nodes of $P$. Let $I$ (the "initial" nodes) be the set of all pairs $(c_0 c_i)$ for all $i$, and let $F$ (the "final" nodes) be the set of all pairs $(c_j c_{m-1})$ for all $j$. Find a directed path $P_q$ from any node in $I$ to any node in $F$ for which the penalty $f(P_q).hi$ is minimum.
3. Concatenate the paths $P_0$, $P_1$, $P_2$, $P_3$ to form a cycle. Let $P$ be the polygon described by that cycle, and let $d$ be the interval $\texttt{max}_q\, f(P_q)$.

4. Start with $H = P$. For each extremal point $t_q$, do the following: Let $a$, $b$ be the current neighbors of $t_q$ in $H$, and let $d'$ be the interval `I_Between` $(t_q, ab)$. If $d'.lo \le \varepsilon$, then remove $t_q$ from $H$, and set $d \leftarrow (\mathtt{max}\{d.lo, d'.lo\}, d.hi + d'.hi)$.
5. Output the polygon $H$, along with the interval $d$, asserting that polygon $H$ is a $(-\varepsilon)$-convex $d$-hull for $S$.

For simplicity, the description of the algorithm includes exact comparison tests in $x$ and $y$. However, these exact tests can be replaced by approximate ones through a straightforward modification of the algorithm (involving a cleanup step to ensure that no two points are too close), at the cost of a small increase in the bound on $d.hi$.

The running time of the algorithm is dominated by step 2e, which can be performed in time $O(|A| \log |A|) = O(n^3 \log n)$ time and $O(|N|) = O(n^2)$ space by a standard graph-theoretic algorithm. (The set of arcs $A$ can be generated on the fly and does not need to be stored explicitly.)

Note that if the accuracy $\lambda$ of all the epsilon boxes used is known in advance, then it suffices to find a feasible path rather than an optimal path in step 2e. In this case the algorithm's complexity is reduced to $O(n^3)$.

Although these bounds appear extravagant, the algorithm can actually be expected to perform quite well in practice because the monotone chains $C_q$ are likely to contain only a small subset of the original points $S$. For points uniformly distributed in a square, Barndorff-Nielsen and Sobel [1] show that the expected size of the chains $C_q$ is only $O(\log n)$. The running time is then dominated by the construction of the chains, which takes $O(n \log n)$ time. (Actually, in our case the analysis is complicated by the imprecise nature of the tests in step 2a; however, a more detailed analysis shows that these imprecise tests can affect the expected size of the chain $C_q$ by at most an additive constant.)

Note that for points uniformly distributed in a square, the preprocessing technique of Golin and Sedgewick [4] can also be used to reduce the set of points to expected size $O(\sqrt{n})$, which reduces the overall expected running time to $O(n)$.

The existence of the paths $P_q$ is guaranteed by the following lemma:

LEMMA 19. *Given a set of points $S$, there exists a convex $2(\varepsilon + \lambda)$-hull $D$ for $S$, with vertices in $S$, such that:*

1. *The polygon $D$ includes the four extremal points $t_0, t_1, t_2, t_3$ of $S$.*
2. *Every point of $S$ not inside $D$ is in the bounding box of some edge of $D$.*
3. *For $0 \le q \le 3$, the xy-monotone chain $D_q = (d_0, \ldots, d_k)$ between $t_q$ and $t_{q+1}$ satisfies*
   a. `E_Pos`$(-\varepsilon, t_q, d_i, t_{q+1}) =$ `true`, *and*
   b. `E_Pos`$(-\varepsilon, d_{i-1}, d_i, d_{i+1}) =$ `true`,
   *for $0 < i < k$.*

PROOF. Let $\sigma = 2(\varepsilon + \lambda)$. Let $Q$ be the quadrilateral defined by the four extremal points $t_0, t_1, t_2, t_3$. For each pair of extremal points $(t_q t_{q+1})$ of $S$, consider the set $S'_q$ consisting of the points $t_q$, $t_{q+1}$, and all points $s$ of $S$ that are more than $2\sigma$ away from $Q$ and such that $Pos(t_q, s, t_{q+1})$. Let $S''_q$ be all points of $S'_q$ that are not

$xy$-dominated by other points in quadrant $q$. By Theorem 18, the set $S''_q$ has a $(-\varepsilon - \lambda)$-convex $\sigma$-hull $D_q$. The polygon $D_q$ must include the extremal points $t_q$ and $t_{q+1}$, since these points are more than $2\sigma$ away from any other point of $S''_q$.

Now consider the polygon $D$ that is the convex hull of the union of the hulls $D_q$. Any consecutive triple in any $xy$-monotone chain of $D$ is a triple of consecutive vertices in some $D_q$ and is therefore $(-\varepsilon - \lambda)$-positive, so the $\lambda$-accurate test $\texttt{E\_Pos}(-\varepsilon, d_{i-1}, d_i, d_{i+1})$ is guaranteed to return $\texttt{true}$. Furthermore, every nonextremal vertex of $D_q$ is more than $2\sigma$ away from $Q$, so

$$\texttt{E\_Pos}(-\varepsilon, t_q, d_i, t_{q+1}) = \texttt{true}$$

by Lemma 4.

By the arguments above, polygon $D$ satisfies all three conditions of the lemma. Furthermore, polygon $D$ is also a $\sigma$-hull for $S$ since every point of $S$ that is not in some $S''_q$ is either inside $D$, or at most $2\sigma$ away from $Q$ or from some polygon $D_q$, both of which are in $D$. □

THEOREM 20. *Algorithm 2 produces a polygon $H$ that is a $(-\varepsilon)$-convex $\delta$-hull for the given set $S$, where $\delta = 6\varepsilon + 7\lambda$.*

PROOF. First, we argue that the $xy$-monotone chains $D_q$ of the polygon $D$, whose existence is proved by Lemma 19, must appear as directed paths in the graphs $G_q$ constructed by our algorithm. Condition 3a of the lemma guarantees that the vertices of $D_q$ will survive in the set $S_q$ constructed in step 2a, and condition 2 guarantees that they will be included in the chain $C_q$ constructed in step 2b. Thus, all edges of $D_q$ will appear as nodes of $G_q$. Furthermore, condition 3b guarantees that every consecutive triple of vertices of $D_q$ appears as an arc of $G_q$. Therefore, step 2e will always be able to find some path $P_q$.

Second, we argue that the polygon $P$ computed in step 3 is a $d$-hull for $S$, where $d$ is the interval computed in that step. Observe that any point $s$ of $S$ that is not inside $P$ must lie in the bounding box of an edge $c_i c_j$ of $P$, because otherwise point $s$ would dominate some vertex of $P$. Moreover, since $P$ is convex, the distance $\delta_s$ between $s$ and $P$ is the same as the distance between $s$ and edge $c_i c_j$. Therefore, by Lemma 4, the penalty of the node $c_i c_j$ computed in step 2d is an interval that contains $\delta_s$. Thus, the interval $d$ computed in step 3 contains the maximum of the distances $\delta_s$ for all points $s$, so $P$ is a $d$-hull for $S$.

Third, we argue that the interval $d$ computed in step 3 is bounded above by the quantity $2\varepsilon + 3\lambda$. Since step 2e finds a path with minimal penalty, the path $P$ must have a penalty no greater than that of the chains $D_q$, which have penalty at most $2(\varepsilon + \lambda)$ by Lemma 19. On the other hand, the penalties assigned to the paths are measured by a $\lambda$-accurate box, so they could be overestimated by at most $\lambda$.

Finally, we argue that the polygon $H$ computed in step 4 is a $(-\varepsilon)$-convex $(4(\varepsilon + \lambda))$-hull for the polygon $P$. For each vertex $v$ of $H$, denote by $h(v)$ the distance between $v$ and the diagonal connecting its two neighboring vertices. By Lemma 9, the polygon $H$ is $(-\varepsilon)$-convex if and only if $h(v) > 2\varepsilon$ for all $v$. At the beginning of step 4, the condition $h(v) > 2\varepsilon$ is satisfied for all vertices $v$ except for the extremal points $t_q$. Moreover, by convexity, deleting a vertex from $H$ cannot decrease the value of $h(v)$ for any other vertex $v$. Since step 4 deletes all vertices for which $h(v) \leq 2\varepsilon$, the final polygon $H$ is $(-\varepsilon)$-convex. On the other hand, step 4 only deletes a vertex $v$ if $h(v) \leq 2(\varepsilon + \lambda)$. Therefore, the new polygon resulting from each deletion is an $(\varepsilon + \lambda)$-hull for the previous one. Since at most four vertices are deleted, the final polygon $H$ is a $(4(\varepsilon + \lambda))$-hull for the starting polygon $P$.

We conclude that the returned polygon $H$ is a $(-\varepsilon)$-convex $\delta$-hull, where $\delta \leq (2\varepsilon + 3\lambda) + 4(\varepsilon + \lambda) = 6\varepsilon + 7\lambda$. □

**Acknowledgments.** The Epsilon Geometry framework was inspired by a discussion with Bernard Chazelle, Herbert Edelsbrunner, Michel Gangnet, Ricky Pollack, Franco Preparata, and Micha Sharir. Some of the key ideas used in the convex-hull algorithm (in particular, the idea to start with *xy*-monotone chains of nondominated points) were suggested by Victor Milenkovic. We would like to thank John Hershberger for this comments on earlier versions of this work. We are grateful to DEC Systems Research Center and the AT&T Foundation for their financial support.

**Appendix. Implementation Results.** The point-inclusion algorithm of Section 4 and the convex-hull algorithm of Section 5.3 have been implemented in Modula-2+ on the Firefly workstation [24] at the DEC Systems Research Center. A package for arbitrary-precision arithmetic [9] was used to verify the correctness of these implementations over a variety of input.

The implementations are built on top of a library of two-dimensional primitives (including *Coincident*, *Collinear*, *Pos*, *Neg*, and *Between*) and interval arithmetic functions (`min`, `max`; $\sqcup$, $\sqcap$, etc.). This library is about 150 lines of code. The point-inclusion algorithm is about 50 lines of code, while the convex-hull algorithm is about 200.

Figure 18 illustrates the different $(-\varepsilon)$-convex $\delta$-hulls constructed by the convex-hull algorithm for three sets of 512 points, as the input parameter $\varepsilon$ is increased from 0.02 to 0.08 to 0.32. The first set of points is uniformly distributed in a square with sides of length 2. The second set is a Gaussian distribution of points on a disk of radius 1. The third set has one-third of its points randomly distributed on a circle of radius 1, one-third of its points randomly distributed on a disk of radius 1, and one-third of its points randomly distributed in very close proximity (between 0.001 and 0.033 units) to the chosen points on the circle.

Below the figures are tables showing the requested value of $\varepsilon$ for each row, the actual values of $\varepsilon$ and $\delta$ (computed using exact arithmetic) for the $(-\varepsilon)$-convex $\delta$-hulls produced, and the value of $\delta$ returned by the algorithm. Note that the convex-hull algorithm actually performs quite well in practice—indeed, in every case considered here, the algorithm actually computes a $(-\varepsilon)$-convex $\delta$-hull with $\delta \leq \varepsilon$.

Table 1 summarizes some performance tests on the convex-hull algorithm. In each test, $(-0.01)$-convex $\delta$-hulls were computed for 20 different sets of $n$ points.

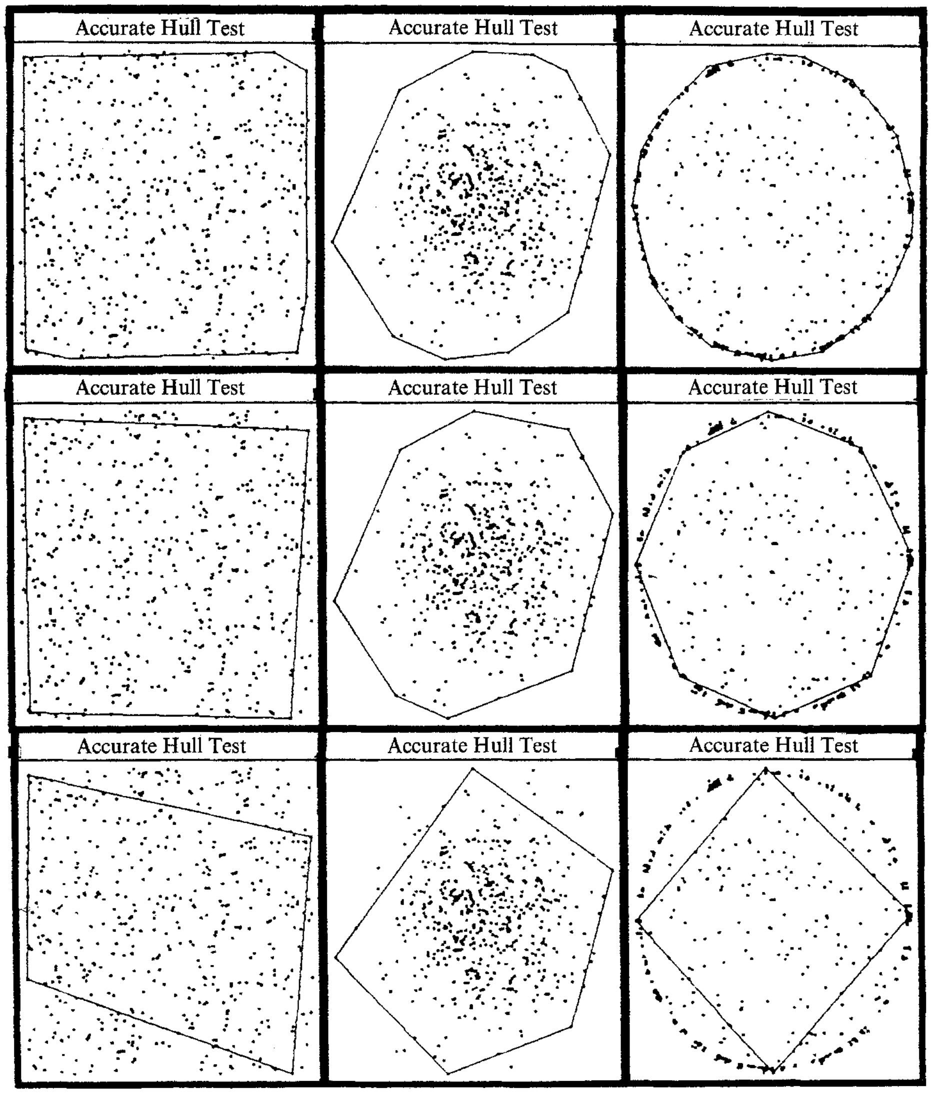

**Fig. 18.** Some approximate hulls.

| ε (requested) | ε (actual) | | | δ (actual) | | | δ (reported) | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.020 | 0.024 | 0.043 | 0.020 | 0.019 | 0.016 | 0.014 | 0.063 | 0.017 | 0.020 |
| 0.080 | 0.631 | 0.103 | 0.129 | 0.055 | 0.075 | 0.051 | 0.171 | 0.075 | 0.079 |
| 0.320 | 0.461 | 0.329 | 0.506 | 0.267 | 0.199 | 0.158 | 0.493 | 0.249 | 0.158 |

**Table 1.** Performance statistics for convex-hull algorithm.

| | Uniform | | Gaussian | | Circular | |
|---|---|---|---|---|---|---|
| $n$ | $t$ | $\lvert C_q\rvert$ | $t$ | $\lvert C_q\rvert$ | $t$ | $\lvert C_q\rvert$ |
| 32 | 1 | 15 | 1 | 12 | 2 | 19 |
| 64 | 3 | 18 | 2 | 14 | 4 | 29 |
| 128 | 5 | 21 | 4 | 16 | 14 | 46 |
| 256 | 9 | 23 | 9 | 19 | 39 | 70 |
| 512 | 18 | 27 | 16 | 21 | 111 | 105 |
| 1024 | 34 | 31 | 33 | 25 | 278 | 144 |
| 2048 | 66 | 32 | 64 | 25 | 544 | 181 |
| 4096 | 130 | 33 | 130 | 30 | 1088 | 227 |
| 8192 | 262 | 38 | 252 | 32 | 1818 | 268 |

The value $t$ gives the total number of seconds spent by the algorithm to compute the 20 hulls. The value $|C_q|$ gives the average number of candidate points in the four monotone subchains constructed for each hull by the algorithm.

The table includes results for uniform, Gaussian, and circular distributions of points, as described above. Note that for uniform and Gaussian distributions of 8000 points or less, the algorithm appears to run in approximately linear time. Note also that the size of the monotone subchains $|C_q|$ appears to be $O(\log n)$ for these distributions, as expected. For points in a circular distribution, the algorithm's performance degrades and appears to be about cubic in the size of the monotone subchains $|C_q|$, which is consistent with the analysis of Section 5.3.

## References

[1] O. Barndorff-Nielsen and M. Sobel, On the Distribution of the Number of Admissible Points in a Vector Sample. *Theory of Probability and Its Applications*, XI(2) (1966), 249–269.

[2] D. Dobkin and D. Silver, Recipes for Geometry and Numerical Analysis—Part I: An Empiricial Study. *Proceedings of the 4th Annual ACM Symposium on Computational Geometry*, 1988, pp. 93–105.

[3] S. Fortune, Stable Maintenance of Point Set Triangulations in Two Dimensions. *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, 1989, pp. 494–499.

[4] M. Golin and R. Sedgewick, Analysis of a Simple Yet Efficient Convex Hull Algorithm. *Proceedings of the 4th Annual ACM Symposium on Computational Geometry*, 1988, pp. 153–163.

[5] D. H. Greene and F. F. Yao, Finite-Resolution Computational Geometry. *Proceedings of the 27th IEEE Symposium on the Foundations of Computer Science*, 1986, pp. 143–152.

[6] L. Guibas, L. Ramshaw, and J. Stolfi, A Kinetic Framework for Computational Geometry. *Proceedings of the 24th IEEE Annual Symposium on Foundations of Computer Science*, 1983, pp. 100–111.

[7] L. Guibas, D. Salesin, and J. Stolfi, Epsilon Geometry: Building Robust Algorithms from Imprecise Computations. *Proceedings of the 5th Annual ACM Symposium on Computational Geometry*, 1989, pp. 208–217.

[8] L. Guibas and J. Stolfi, CS445 Computational Geometry Lecture Notes, Computer Science Department, Stanford University, Winter 1983.

[9] J. C. Herve, F. Morain, D. Salesin, B. P. Serpette, J. Vuillemin, and P. Zimmermann, BigNum: A Portable and Efficient Package for Arbitrary-Precision Arithmetic. Research Report No. 1016,

Institut National de Recherche en Informatique et en Automatique (INRIA), Rocquencourt, 1989.

[10] C. Hoffman, The Problems of Accuracy and Robustness in Geometric Computation. *Computer*, **22** (1989), 31–42.

[11] C. M. Hoffman, J. E. Hopcroft, and M. S. Karasick, Towards Implementing Robust Geometric Computations. *Proceedings of the 4th Annual ACM Symposium on Computational Geometry*, 1988, pp. 106–117.

[12] J. W. Jaromczyk and G. W. Wasilkowski, Numerical Stability of a Convex Hull Algorithm for Simple Polygons. Technical Report No. 177-90, University of Kentucky, 1990.

[13] D. E. Knuth, Axioms and Hulls. Manuscript, Stanford University, 1991. To appear as a Springer-Verlag Monograph.

[14] V. J. Milenkovic, Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic. *Artificial Intelligence*, **37** (1988), 377–401.

[15] V. J. Milenkovic, Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic. Ph.D. thesis, Carnegie-Mellon, 1988. Available as CMU Report CMU-CS-88-168.

[16] V. J. Milenkovic, Calculating Approximate Curve Arrangements Using Rounded Arithmetic. *Proceedings of the 5th Annual ACM Symposium on Computational Geometry*, 1989, pp. 197–207.

[17] V. J. Milenkovic, Double Precision Geometry: A General Technique for Calculating Line and Segment Intersections Using Rounded Arithmetic. *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, 1989, pp. 500–505.

[18] V. J. Milenkovic and Z. Li, Constructing Strongly Convex Hulls Using Exact or Rounded Arithmetic. *Proceedings of the 6th Annual ACM Symposium on Computational Geometry*, 1990, pp. 235–243.

[19] T. Ottmann, G. Thiemt, and C. Ullrich, Numerical Stability of Geometric Algorithms. *Proceedings of the 3rd Annual ACM Symposium on Computational Geometry*, 1987, pp. 119–125.

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

[21] D. Salesin, Epsilon Geometry: Building Robust Algorithms from Imprecise Computations. Ph.D. thesis, Stanford University, 1991. Available as Report STAN-CS-91-1398, Stanford, CA.

[22] M. Segal and C. Séquin, Consistent Calculations for Solids Modeling. *Proceedings of the 1st Annual ACM Symposium on Computational Geometry*, 1985, pp. 29–38.

[23] K. Sugihara and M. Iri, Geometric Algorithms in Finite-Precision Arithmetic. Research Memorandum RMI 88-10, University of Tokyo, September 1988.

[24] C. P. Thacker, L. C. Stewart, and E. H. Satterthwaite, Jr., Firefly: A Multiprocessor Workstation. Research Report no. 23, DEC Systems Research Center, Palo Alto, CA, 1987.