Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Discrete Algorithms 3 (2005) 1–26
JOURNAL OF DISCRETE ALGORITHMS
www.elsevier.com/locate/jda

# Optimizing a constrained convex polygonal annulus

Gill Barequet [a,*,1], Prosenjit Bose [b,2], Matthew T. Dickerson [c,3], Michael T. Goodrich [d,4]

[a] *Center for Graphics and Geometric Computing, Department of Computer Science, The Technion—Israel Institute of Technology, Haifa 32000, Israel*
[b] *School of Computer Science, Carleton University, Herzberg Room 5302, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6*
[c] *Department of Mathematics and Computer Science, Middlebury College, Middlebury, VT 05753, USA*
[d] *Center for Algorithm Engineering, Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA*

Available online 3 February 2004

**Abstract**

In this paper we give solutions to several constrained polygon annulus placement problems for offset and scaled polygons, providing new efficient primitive operations for computational metrology and dimensional tolerancing. Given a convex polygon $P$ and a planar point set $S$, the goal is to find the thinnest annulus region of $P$ containing $S$. Depending on the application, there are several ways this problem can be constrained. In the variants that we address the size of the polygon defining the inner (respectively, outer) boundary of the annulus is fixed, and the annulus is minimized by minimizing (respectively, maximizing) the outer (respectively, inner) boundary. We also provide solutions to a related known problem: finding the smallest homothetic copy of a polygon containing a set of points. For all of these problems, we solve for the cases where smallest and largest are defined by either the offsetting or scaling of a polygon. We also provide some experimental results from implementations of several competing approaches to a primitive operation important to all the above variants: finding the intersection of $n$ copies of a convex polygon.
© 2003 Elsevier B.V. All rights reserved.

* Corresponding author.
*E-mail addresses:* barequet@cs.technion.ac.il (G. Barequet), jit@scs.carleton.ca (P. Bose), dickerso@middlebury.edu (M.T. Dickerson), goodrich@acm.org (M.T. Goodrich).

[1] Work of this author was supported in part by the Fund for the Promotion of Research at the Technion and by a Fialkow Academic Lectureship.
[2] Work of this author was supported by the Natural Science and Engineering Research Council of Canada under grant OGP0183877.
[3] Work of this author was supported by NSF Grants CCR-9902032 and CCR-9732300.
[4] Work of this author was supported by NSF Grants CCR-9732300 and PHY-9980044.

1570-8667/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2003.12.004

*Keywords:* Offset; Annuli; Tolerancing; Optimization

## 1. Introduction

The research areas of computational metrology and dimensional tolerancing are focused on developing repertories of basic tests, such as for "roundness", "flatness", and angle conformity, so as to build a systematic collection of efficient methods for determining if manufactured parts conform to their design specifications [29,33]. After a part is manufactured, its surface is sampled by a device known as a coordinate measuring machine (CMM) and these sampled points are then tested against various design constraints to see if this part is conforming or not. The collection of tests that can be done simply and efficiently is therefore a limiting factor on the richness and sophistication of the constraints that designers can specify with confidence that their designs will be faithfully tested. Efficient methods for several computational-metrology primitives have been presented in the algorithms and computational-geometry literatures (see, e.g., [1,8,10,12,14–16,24,25,27, 30–32,34]). This paper provides methods for testing how well a set of points matches the boundary of a convex polygon.

### *1.1. Offset polygons and their properties*

Computing optimal placements of annulus regions is a fundamental aspect of many computational metrology tests for quality control in manufacturing. For example, the width of the thinnest circular annulus containing a set of points is the measure used for testing "roundness" by the American National Standards Institute (see [17, pp. 40–42]) and by the International Standards Organization. The usual goal is to find, for a certain type of annulus region, a placement of the annulus that contains a given set or subset of points. Optimality of the placement can be measured either by *minimizing* the size of the annulus region necessary to contain all (or a certain number) of the points, or by *maximizing* the number of points contained in a fixed-size annulus. In addition to the tolerancing applications, these problems also arise in pattern matching and robot localization [18]. Thus we are interested in extending the collection of simple and efficient tolerancing tests to include new kinds of minimum or maximum annulus placement constraints.

One set of such problems studied recently by Barequet et al. in [5] involves the optimal placement of *polygonal* annulus regions. These authors noted that the polygonal annulus can be defined as the difference region between two *scaled* concentric copies or two *offset* copies of a convex polygon $P$. The scaled polygons correspond to the convex distance functions (also called Minkowski functionals [20, p. 15]), which are extensions of the notion of scaling circles (in the Euclidean case) to convex polygons. There have been several papers (e.g., [13,23,26]) which explore the Voronoi diagram based on these distance functions.

Let us define formally the offset of a convex polygon. A convex polygon $P$ is the intersection of a collection of closed halfplanes $\{H_i\}$, each defined by an edge of $P$. The offset polygon is the intersection of $\{H_i(\varepsilon)\}$, where $H_i(\varepsilon)$ is the halfplane parallel to $H_i$

with bounding line translated by distance $\varepsilon$. Positive (respectively, negative) values of $\varepsilon$ stand for translating the edges outward (respectively, inward) from the polygon. It is well known that the offset operation moves the polygon vertices along the *medial axis* of the polygon, so that an edge "disappears" when two polygon vertices meet on a medial-axis vertex. (This happens only when the polygon is offset inward.) We denote throughout the paper the medial axis of a polygon $P$ by $\mathrm{MA}(P)$.

This offset operation was studied by Aichholzer et al. [3,4] in the context of a novel polygon skeleton, called the *straight skeleton*. They discussed the straight skeleton for both convex and simple polygons. For convex polygons the straight skeleton is just the well-known medial axis. For nonconvex simple polygons, however, the straight skeleton and the medial axis are different. In this paper we deal with convex polygons only, and we will refer to the offset skeleton as the polygon medial axis.

Barequet et al. [6] also studied the polygon-offset operation in a different context: that of a new distance function and its related Voronoi diagram. They give efficient algorithms for computing compact representations of the nearest- and furthest-neighbor diagrams. Polygon offsets were also used in the solution to various annulus placement problems [5]. In many applications, including those dealing with manufacturing processes, defining distance in terms of an offset from a polygon (either inward or outward) is more natural than scaling. This preference for offsetting is motivated by the fact that the absolute error of a production tool (milling head, laser beam, etc.) is independent of the location of the produced feature relative to some artificial reference point (the origin). Thus, a tool is more likely to allow (and expect) local errors bounded by some tolerance, rather than scaled errors relative to some (arbitrary) center. For this reason, a study of the polygon-offset operation, of the related distance function and its Voronoi diagram, and of annulus-placement problems for offset polygons, are particularly interesting. Theoretical aspects of this distance function and Voronoi diagram were studied in [6], and were used in that paper as well as in [5] in solutions to the offset versions of several problems involving one or the other definitions of optimization given above.

### *1.2. Related results*

In [5] several annulus placement problems are solved based on offset polygons. Most of the problems involved optimizing the placement by fixing $\delta$ (the width of the annulus region) and maximizing the number of points contained. Algorithms are given for convex and for simple polygons, and for translation only, as well as for a general placement with translation and rotation allowed. In that paper the authors also solve some related decision problems including an on-line variant. It is suggested that this approach to annulus placement may provide a robust solution to various problems that arise in robot localization, with the presence of "noisy data" that need not be contained in the best placement.

As was noted earlier, a related optimization problem is fixing the minimum number of points to be contained (sometimes the entire set) and minimizing the size of the annulus needed to contain those points. One approach to minimizing the annulus is to apply a constraint that either the inner or outer boundary of the annulus is of fixed size. Fixing the size of the inner or outer boundary of an annulus is itself an important aspect of quality control. Consider, for example, manufacturing a part (like a cylinder) that must fit inside

a corresponding manufactured sleeve. For the part that must fit inside the sleeve, the outer boundary defining the annulus has an absolute maximum which is fixed. For the sleeve itself, however, it is the inner boundary that is crucial and must be fixed.

For the case where the annulus boundary is circular, it is shown in [8] that when the inner or outer circle is of fixed size, the placement problem can be solved more efficiently than for the general annulus minimization problem. Likewise, we seek a placement of a *polygonal* annulus region that contains all points, but we follow the same idea of fixing one boundary of the annulus and minimizing the width (or tolerance) of the region by offsetting or scaling the other boundary. In particular, we extend the approach of [8] to polygonal annulus regions, and also present some new approaches.

### *1.3. Problem statements*

Throughout this paper we will have a set $S$ of $n$ points and a convex polygon whose complexity is $m$. We will omit the definitions of $m$ and $n$ wherever they are clearly understood from the context.

In this paper we explore a new set of convex-polygon annulus placement problems where one of the two annulus boundaries (inner or outer) is fixed. (By a "placement" we mean a *translation* of the annulus, where no rotation is allowed.) We solve problems for polygonal annulus regions defined for both the polygon-offset and the regular convex polygon scaling distance function. In particular, we give algorithms for the following problems:

**Problem 1.** Given a convex polygon $P$ and a set of points $S$, find the largest possible polygon $P^*$—an inner offset (or a scaled version) of $P$—and a placement of the annulus defined by $P$ and $P^*$, such that all the points in $S$ are covered by the annulus.

**Problem 2.** Given a convex polygon $P$ and a set of points $S$, find the smallest possible polygon $P^*$—an outer offset (or a scaled version) of $P$—and a placement of the annulus defined by $P$ and $P^*$, such that all the points in $S$ are covered by the annulus.

Fig. 1(a) shows a sample polygon $P$ (solid) and an outer offset of it (dashed). For reference, $\mathrm{MA}(P)$ (the medial axis of $P$) is also shown in light lines. Fig. 1(b) shows a set of points $S$. Fig. 1(c) shows a solution to Problem 2 for the given $P$ and $S$. That is, we see an annulus region containing all the points of $S$, whose fixed inner boundary is $P$ and whose outer boundary is the smallest possible offset of $P$ such that $S$ can be contained in the annulus.

The following problem can be viewed as a special case of Problem 2, when the inner polygon (the polygon defining the inner boundary of the annulus) is null, or as a variant of the famous "smallest enclosing circle" problem for convex polygons:

**Problem 3.** Given a convex polygon $P$ and a set of points $S$, find the smallest offset (respectively, scaled) translation of $P$ containing all the points in $S$.

A substep of several approaches to the above problems is the following:

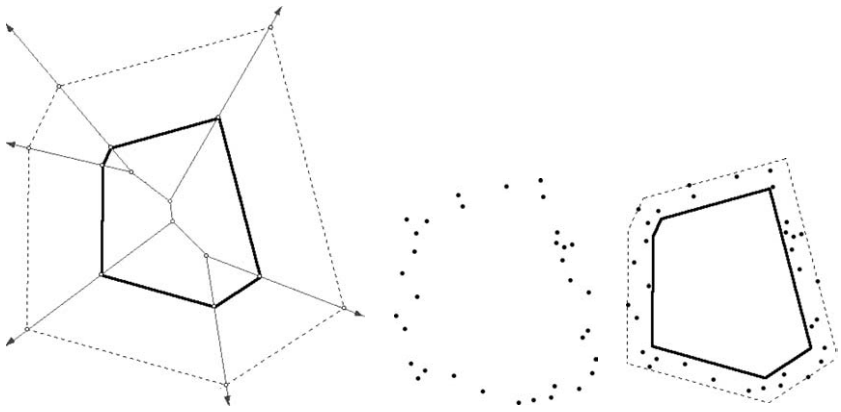



Fig. 1. An optimal offset annulus and a placement for covering a set of points. (a) Polygon and its medial axis. (b) Points. (c) Covering annulus.

**Problem 4.** Given a convex polygon $P$ and a set $S$ of $n$ translations, compute the intersection of the $n$ translated copies of $P$.

### *1.4. Overview of results*

We first present some approaches to Problem 4, which is an important geometric primitive in several of our algorithms. We then provide subquadratic-time algorithms for Problems 1 and 2 which are nontrivial extensions of ideas found in [8] from circular annuli to polygonal annuli. In particular, we give general solutions that solve both the scaled and offset versions of the problems. Our algorithm for Problem 1 requires $O((n+m)\log(n+m))$ time for scaling, and $O(n(\log n + \log^2 m) + m(\log n + \log m))$ time for offsetting, where $n$ is the number of points in the set $S$ and $m$ is the complexity of the polygon $P$. Our first algorithm for Problem 2 requires $O(nm(\log n + \log m))$ time (for either scaling or offsetting).

We then present two algorithms for Problem 3, finding the smallest enclosing polygon. One approach requires $O(n \log m + m)$ expected time for scaled polygons and $O(n \log^2 m + m)$ expected time for offsetting, and is an extension to convex polygons of the well-known randomized incremental algorithm for finding the smallest enclosing circle [36]. The second method is a new approach based on the medial axis of a polygon and on our solution to Problem 4, and can be implemented to run either in $O(n \log h + m)$ time or in $O(nm)$ time (both in the worst case), where $h$ is the complexity of the convex hull of $S$. We then extend our solution to Problem 4 to provide a different solution to Problem 2 which is simpler than our first algorithm and requires only $O(n \log n(\log n + \log m) + m)$ time—an order of magnitude improvement when $m$ and $n$ are comparable.

The paper is organized as follows. In Section 2 we provide some preliminary observations and properties of offset polygons. In Section 3 we present several approaches to computing the intersection and union of several copies of the same convex polygon. In Sections 4 and 5 we fix either the inner or outer boundary of the annulus, and minimize its

width. In Section 6 we investigate further the problem of minimizing the polygon enclosing a given set of points. In Section 7 we describe an alternative algorithm for minimizing the annulus with a fixed inner boundary. We end in Section 8 with some concluding remarks.

## 2. Preliminary observations

In this section we present both some terminology and some additional properties of the polygon-offset distance function. We first define what we mean by a *placement* of a polygon. Throughout the paper we assume that each polygon has a fixed reference point. For scaled polygons it is natural to assume the center of scaling is contained in the polygon. For offset polygons the natural reference is the offsetting center: the point to which the inner polygon collapses when the polygon is offset inward. (This point is the center of the medial axis of the polygon [6].) In other words, this point is the center of the largest circle contained in the polygon. In degenerate cases the offset center can be a segment, so we arbitrarily select a point in it as the center. (The median of the segment is an appropriate choice.) By translating a copy of the polygon $P$ to some point $q$, we mean the translation of $P$ that maps its center (reference point) to $q$. Similarly, when we speak of the *reflection* of $P$, we mean the rotation of $P$ by $\pi$ around its center point. The translation of a reflection of $P$ to a point $q$ translates the polygon so that the center of the reflected copy is mapped to $q$.

We now make some observations about the polygon-offset operation, and the related distance function and Voronoi diagrams. These observations are analogous to well-known facts about other distance functions, in particular the Euclidean distance function. We include them here, however, because it is not obvious that these properties that hold true for Euclidean distance also hold for offset distance. One reason it is not obvious is that the polygon-offset distance function is not a metric [6]. In fact, like the more common Minkowski functions (scaled polygon distance), it is not even symmetric. It is proven in [6] that the polygon-offset distance function does not satisfy the triangle inequality, and in fact, for collinear points it satisfies a reverse inequality. Also, whereas the Voronoi regions for the Minkowski functions are always star-shaped, there exist point sets for which the Voronoi diagram based on the polygon-offset distance function have non-star-shaped regions. Fig. 2 shows such an example: a quadrilateral (shown with its medial axis) defines a convex-offset distance function; the Voronoi cell of one point of a 3-point set is non-star-shaped.

Some of our algorithms make use of Voronoi diagrams based on the scaled (Minkowski) or offset distance functions. In both cases the *bisector* of two points $p, q$ is (in the nondegenerate case) the polyline that contains all points equidistant from $p$ and $q$.[5] That is, it is the set of points $x$ such that $d(p, x) = d(q, x)$. Note that we could also define it symmetrically as the set of points $x$ such that $d(x, p) = d(x, q)$, which matches the first definition when we reflect the underlying polygon. Since neither distance function is symmetric, these

[5] This is a single connected polygonal curve with two rays at its ends, separating the plane into two connected regions, each containing one of the two points.

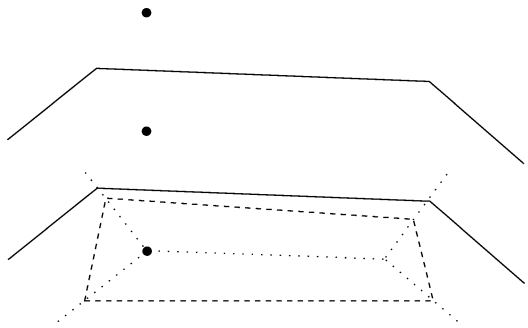


Fig. 2. A non-star-shaped Voronoi cell of a point (for the offset distance function).

two definitions result in different bisectors. However, all of the following observations and lemmas hold regardless of which definition is used.

**Observation 1** (Scale and offset). *The bisector between two points $p$ and $q$ has a* median *segment $s$ that contain all points that are equidistant from $p$ and $q$ and such that this distance is the minimum of all points on the bisector. In both directions along the bisector away from $s$, distances from $p$ and $q$ to points on the bisector are monotonically nondecreasing.*

This can be seen by examining the pair of smallest offset (or scaled) touching copies of the polygon $P$ placed at the points $p$ and $q$. Since the polygons are convex, the intersection is a segment or a single point. As the polygons further grow outward, the median (point or segment) is completely contained in the intersection of any larger copies. Note that the median of a bisector of $p$ and $q$ is analogous to the midpoint of the segment $pq$ in the Euclidean distance function. In case the defining polygon does not have parallel edges, the median is always a single point.

We may use the same idea to show the following:

**Observation 2** (Scale and offset). *Given a point $p$, a line $\ell$, and a point $q$ on $\ell$, there exists a direction along $\ell$ from $q$ such that $d(x, p) \geqslant d(q, p)$ for all points $x$ on $\ell$ in that direction.*

Note that this is true for $d(p, x) \geqslant d(p, q)$ as well as for $d(x, p) \geqslant d(q, p)$, but the directions might be different!

### *2.1. Feasible regions of placement*

We now present more observations and lemmas that will be used in our algorithms. In particular, following the approach of [8], we aim to bound the region in which the fixed-size polygon (that is, its center) can be placed. For Problem 1 we seek the possible placements of the fixed-size outer polygon that contain all the points, and for Problem 2 we seek the possible placements of the fixed-size inner polygon, that do not properly contain

any of the points. We denote these regions as the sets of "feasible placements." The following observations are well-known and have been used in several algorithms for different polygon-placement problems [5].

**Observation 3.** *Given a polygon $P$ and two points $p$ and $q$, a translation of $P$ to $p$ contains $q$ if and only if a translation of the reflection of $P$ to $q$ contains $p$.*

This observation follows from simple vector arithmetic and leads to the following generalizations:

**Observation 4.** *A translation of a polygon $P$ to a point $q$ contains all the points of a set $S$ if and only if the intersection of the $n$ copies of the reflection of $P$ translated to the points of $S$ contains $q$.*

**Observation 5.** *A translation of a polygon $P$ to a point $q$ is empty of points from a set $S$, i.e., properly contains none of the points of $S$, if and only if $q$ is not properly contained in the union of $n$ copies of the reflection of $P$ translated to the points of $S$.*

Based on Observation 4, we define a *feasible region* for placements of the annulus region in Problem 1. The feasible region of placements is given by the intersection of $n$ reflected copies of $P$ translated to each of the points in $S$. This feasible region, according to Observation 4, contains all possible placements where the fixed outer polygon contains all the points in $S$. The goal then becomes to find the largest inner polygon (scaled or offset) that can be placed inside this region without containing any point of $S$. If the feasible region of the outer polygon is already empty, then there is no solution at all. A solution to Problem 4 thus provides us with the feasible region.

There is an analogous idea for Problem 2, where we are interested in finding a placement of the inner polygon such that it does not contain any point of $S$. Based on Observation 5, we define a different *feasible region* for placements of the annulus region in Problem 2. The feasible region of placements is given by the complement of the union of $n$ reflected copies of $P$ translated to each of the points in $S$, plus the boundary edges of the region. This feasible region consists of all possible placements where the fixed inner polygon does not properly contain any of the points in $S$. The goal then becomes to find the smallest outer polygon (scaled or offset) that can be placed inside this region while containing all points in $S$.

## 3. Computing intersections and unions of $n$ copies of a convex $m$-gon

In this section we describe several alternative approaches to solving Problem 4, computing the intersection of $n$ translated copies of a convex polygon. Since we present six competing approaches to the same problem, we provide an experimental comparison between five of these approaches. (Although it is asymptotically fast, the Voronoi-diagram approach is impractical and was not included in the experiment.) We also discuss computing and representing the union of $n$ copies of a convex polygon, both in explicit and

compact forms. As mentioned above, solutions to these two subproblems are important primitives in many polygon placement algorithms.

### 3.1. Intersections

It is shown in [7] how the prune-and-search technique of Kirkpatrick and Snoeyink [22] can be used for efficiently finding the intersection points of two translated copies of a convex polygon. We now describe several ways to compute all the vertices of the polygon that is the intersection of $n$ translations of some input polygon with $m$ vertices. These algorithms use well-known techniques but are included here for completeness as we have not previously seen them applied to this problem. There are several other competing approaches as well which we do not outline here. The resulting running times of the following approaches are $O(nm)$, $O(n \log h + m)$ (where $h$ is the complexity of the convex hull of the point set), or $O(n(\log n + \log m) + m)$. The third approach is always asymptotically inferior to the second approach. We have implemented five versions of these algorithms, which we describe in Appendix A.

#### 3.1.1. Brute force 1

A "brute force" approach is to start with two copies of the polygon, and compute their intersection using any of several algorithms for intersecting convex polygons in time which is linear in the total number of vertices. Each of the remaining $n - 2$ polygons can then be iteratively intersected with the polygon resulting from the previous step. Let $P_1, \ldots, P_n$ be our set of $n$ polygon translations. This algorithm is represented as follows:

(1) $P^* := P_1$;
(2) **for** $i := 2, \ldots, n$ **do** $P^* := P^* \cap P_i$;

The important observation is that after each step the resulting intersection is still a convex polygon with at most $m$ edges, each of which is parallel to an edge of the original polygon. Thus each step requires $O(m)$ time for a total of $O(nm)$. This brute force approach is not only simple, but is linear in $n$, and thus is asymptotically efficient when $m$ is small. Note that the same running time can be obtained by repeatedly pairing the polygons and merging.

#### 3.1.2. Brute force 2

A second brute force approach relies on a simple observation. Each edge $e_i$ in the output polygon $P^*$ is determined by a single translated polygon from the input set—in particular, by that polygon which is extremal in the direction orthogonal to $e_i$ and toward the opposite side of the polygon from $e_i$. The algorithm iterates through the $m$ edges of $P$, and for each edge $e_i$ determines in $O(n)$ time which of the $n$ input polygons might contribute an edge $e_i$ to the output by finding extremal points in $S$. We iteratively construct $P^*$, adding one edge at a time and eliminating edges that are cut off. Note that the addition of a single new edge may eliminate more than one edge. If edges are added in rotational order, then the cost of adding the edge $e_i$ is $O(1 + c_i)$, where $c_i$ is the number of earlier edges removed by $e_i$ (possibly $c_i = 0$). Since the total number of edges is at most $m$, the sum of the $c_i$'s is

also $m$. In some sense, this second approach reverses the roles of the inner and outer loops from the previous algorithm. The running time of this approach is also $O(nm)$.

*3.1.3. Using the convex hull*

We can modify both approaches if we make use of the following lemma, and more importantly the corollary.

**Lemma 6.** *A convex polygon contains all points in a set $S$ if and only if it contains all vertices of the convex hull of $S$.*

**Corollary 7.** *The intersection of $n$ translated copies of a polygon $P$ placed at each of $n$ points from a set $S$ is the same as the intersection of $h$ translated copies of $P$ placed at the $h$ vertices of the convex hull of $S$.*

Corollary 7 tells us that we can eliminate non-hull points in a preprocessing step. Depending on the relationship between $h$ (the complexity of the convex hull of $S$) and $m$, the speed-up in the latter computation may pay for the preprocessing cost of computing the convex hull. We compute the convex hull of $S$ in $O(n \log h)$ time [11,21]. Applying the first brute force approach only to the $h$ hull points, we get a total running time of $O(n \log h + hm)$ which is an improvement if $m = \omega(\log h)$ and $h = o(n)$.

The second brute force approach can be improved even further. Given the convex hull of a set of points, we can compute in rotational order the extremal points in directions orthogonal to each of the $m$ edges of $P$ in $O(m + h)$ time by using the "rotating-calipers" method [35]. The output polygon $P^*$ is still constructed one edge at a time. As in the previous method, the overall number of eliminated edges is $m$. The overall running time, including computing the convex hull of $S$, is thus $O(n \log h + m)$.

*3.1.4. A furthest-neighbor Voronoi diagram approach*

A final approach makes use of the furthest-neighbor Voronoi diagram to compute the intersection of the $n$ polygons. First we compute a compact representation of the furthest-neighbor Voronoi diagram of the $n$ points. For convex distance this is done by the algorithm of [26] in $O(n(\log n + \log m) + m)$ time. For polygon-offset distance this could be done [6] slightly slower in $O(n(\log n + \log^2 m) + m)$ time, but as we mention below we can use the first diagram for both distance functions.

Now, we follow the first step (out of three steps) of Lemma 2.1 of [16, p. 124], which constructs the intersection of $n$ congruent circles in $O(n)$ time. Specifically, in [16] the authors find the portion of the intersection of the circles in each cell of the furthest-neighbor Voronoi diagram. They do that by a simple walking (on the diagram) method. All we need to observe is that they amortize the number of jumps between cells of the diagram, and obtain (for the circles case) an $O(n)$ time bound due to the complexity of the diagram. For both our distance functions we do this in $O(n \log m)$ time: from the compact representation of the diagram we need to explicitly compute only the portions that belong to the intersection. This happens $O(n)$ times, and for each we spend $O(\log m)$ time. The complexity

of the accumulated output is $O(n \log m)$. In total we have $O(n(\log n + \log m) + m)$- and $O(n(\log n + \log^2 m) + m)$-time algorithms for the scaling and offsetting distance functions, respectively.

Since we are interested in the intersection of copies of the *original* polygon, to which the unit scale or offset[6] identify, it does not matter which distance function is used. So we may prefer to choose the respective Voronoi diagram of the scaling function, which provides a slightly faster algorithm. The Voronoi diagram approach may also be modified with the precomputation of the convex hull. However, the result is asymptotically slower than the second brute force approach when the convex hull is used.

### 3.2. Unions

A representation of the union of $n$ translated copies of a polygon is also needed, as its complement defines another feasibility region.

Since translated copies of the same convex polygon together define a set of pseudo-disks, the union of $n$ translated copies of a convex $m$-gon has complexity $O(nm)$, where "complexity" refers to the total number of edges (and vertices) possible on its boundary (Kedem et al. [19]). The union can be computed in $O(nm(\log n + \log m))$ time by using a plane-sweep approach.

However, the complexity of the union is only $O(n)$ in terms of the number of polygonal arcs (portions of the original polygon $P$) and intersection points, and thus it may be stored more compactly using an implicit representation. Consider the boundary of a convex polygon $P = (e_0, e_1, \ldots, e_{m-1})$ as being defined by a set of $m$ edges listed in counterclockwise order. We can represent any continuous portion of the boundary of $P$ as $(p, i, q, j)$, where $p$ is the starting endpoint of the polygonal "arc", $e_i$ is the edge containing $p$, $q$ is the terminating endpoint of the polygonal "arc", and $e_j$ is the edge containing $q$. If we consistently represent maximal continuous portions of copies of a convex polygon $P$ in this way, then the bound of [19] regarding pseudo-disks implies that such a *compact representation* of the union of $n$ translated copies of a convex $m$-gon can be stored in $O(n + m)$ space.

Moreover, by a simple divide-and-conquer algorithm, we can construct such a compact representation in much less time than that required for the explicit representation. After preprocessing $P$ in $O(m)$ time (to be able to do intersection tests between translated copies of the convex polygon), we have $O(\log n)$ steps in which we unite two intermediate unions $U_1, U_2$ bounded by $O(n)$ polygonal arcs maintained in sorted order. From [19] we know that $U_1, U_2$ intersect $O(n)$ times. We invoke a plane-sweep procedure in the merge step of $U_1$ and $U_2$. Each time we compare two polygonal arcs it takes $O(\log m)$ time to determine if they intersect in zero, one, or two points. Thus, the merge takes $O(n(\log n + \log m))$ time, and the entire procedure requires $O(n \log n(\log n + \log m) + m)$ time.

[6] We use the term "unit offset" instead of "zero offset", since we normalize the offset operation so that the 0-offset makes the polygon shrink to its center, and the 1-offset leaves it unchanged.

## 4. Minimizing the annulus for a fixed outer polygon

### *4.1. The underlying theorem*

We now address the problem of minimizing an annulus region by fixing the outer polygon and maximizing the inner polygon (Problem 1). We solve this problem for both scaled and offset polygons. In the following theorem and corollary, upon which our algorithm is based, the Voronoi diagram is for the appropriate distance function: either scaled (Minkowski) or offset polygon.

**Theorem 8.** *Given a convex feasible region of possible translations of a polygon $P$, there exists a largest* (*scaled or offset*) *empty polygon* (*properly containing no points in $S$*) *that is centered on one of the following points*:

1. *On a vertex of the nearest-neighbor Voronoi diagram of $S$*;
2. *On an intersection of an edge of this diagram with the boundary of the feasible region*;
3. *At a vertex on the boundary of the feasible region.*

**Proof.** Assume for contradiction that the minimum polygon annulus region is placed in the feasible region (ensuring that all the points are contained in the outer polygon), but not in any of the three possibilities listed in the stated theorem. That is, assume that the center of the polygon is placed either inside a Voronoi region or at a point on a Voronoi edge (that is not a Voronoi vertex), but not on the boundary of the feasible region. We show that the inner polygon could then be enlarged, thus shrinking the size of the annulus region and contradicting the assumption. Suppose the polygon is placed at a point $x$ inside a Voronoi region of a point $q \in S$ and not on the boundary of the feasible region. This implies that $q$ is the nearest neighbor of $x$, and the definition of our distance function further implies that the maximum inner polygon defining the annulus region has $q$ and no other point in $S$ on its boundary. It is thus possible to move $x$ in some direction farther away from $q$ without $x$ becoming closer to any other point in $S$ than it is to $q$. In particular, place an offset (scaled) copy of $P$ at $x$ sized to be tangent to $q$; moving $x$ in the direction orthogonal to the edge of $P$ tangent to $q$ and away from $q$ increases the distance from $x$ to its nearest neighbor $q$, and in doing so increases the size of the maximum polygon placed at $x$ containing no points in $S$, giving us a contradiction. Suppose next that the point $x$ is inside a Voronoi region, but on the boundary of the feasible region. If it is on a vertex of the polygon defining the feasible region, we are in case 3. If it is on an edge of the feasible region, then by Observation 2 we can again move $x$ in some direction farther from its nearest neighbor $q \in S$, and as in the previous case, we have a contradiction.

Suppose, finally, that the best placement $x$ is on an edge of the nearest-neighbor Voronoi diagram. Recall that an edge of the Voronoi diagram between the cells of $p$ and $q$ is part of the bisector between $p$ and $q$ defined by the appropriate distance function. By Observation 1 there is a median point or segment that is equidistant from $p$ and $q$ and contains the closest points to $p$ and $q$ of all such equidistant points. If the median is a point, then there is some direction that $x$ can move along the bisector away from that median point,

that will increase its distance to its nearest neighbors, and thus increase the size of the largest possible polygon not containing any points. The only thing that would prevent the movement of $x$ along this bisector is if $x$ also sits on the boundary of the feasible region, in which we are in case 2 and the theorem holds. In the degenerate case, when the median is a segment and not a unique point, and $x$ is on this segment, we can move $x$ along this median segment such that its distance to its nearest neighbors in $S$ does not increase. We continue either until we reach the end of the median segment and the distance begins to increase (which is a contradiction) or until we reach a Voronoi vertex and we are in case 1, or until we reach a boundary of the feasible region, in which case we are in case 2. □

Hence, we are able to characterize what constrains the inner annulus boundary in this optimization problem:

**Corollary 9.** *The optimal placement of the annulus region, when its outer boundary is fixed, has at least three contact points between the set $S$ and the inner or outer boundary of the annulus region, at least one of which is in contact with the inner boundary* (*the maximized inner polygon*).

### *4.2. The algorithm*

The algorithm for solving Problem 1 is based on Theorem 8. First we construct the feasible region. This is done by computing the intersection of $n$ convex polygons of complexity $m$, which we do in $O(nm)$ or $O(n \log h + m)$ time (see Section 3.1). Next we construct the nearest-neighbor Voronoi diagram of $S$ with respect to the polygon $P$ and the appropriate (scale or offset) distance function. Compact representations can be computed in $O(n(\log n + \log m) + m)$ time (for the scaling case) or in $O(n(\log n + \log^2 m) + m)$ time (for the offsetting case). Finally, we check a discrete set of at most $n$ Voronoi vertices, $2n$ intersections between Voronoi edges and the convex feasibility region (the farthest from the medians of the edges), and $m$ vertices of the feasible region, to find which allows the maximal polygon.

For a Voronoi vertex we can test containment in the feasibility region in $O(\log m)$ time. For a Voronoi edge we can find intersections with the feasibility region in $O(\log m)$ time. To find the maximal inner polygon, we just need to know the distance to the nearest neighbor in $S$. For Voronoi vertices and edges, this is known. For vertices of the feasibility region we can do point location in the compact Voronoi diagram in $O(\log n + \log m)$ time, and computing the actual distance requires additional $O(\log m)$ time. The total running time for checking the $O(n + m)$ possible locations is therefore $O(n \log m + m(\log n + \log m))$ time.

**Theorem 10.** *The minimum polygon annulus with a fixed outer polygon can be computed in $O((m+n)\log(m+n))$ time* (*for scaling*) *or in $O(n(\log n + \log^2 m) + m(\log n + \log m))$ time* (*for offsetting*).

## 5. Minimizing the annulus for a fixed inner polygon

### *5.1. The underlying theorem*

In this section we address the problem of minimizing an annulus region by fixing the inner polygon and minimizing the outer polygon (Problem 2). As in the previous sections, the algorithms work both for the scaled and offset polygons. In what follows, the furthest-neighbor Voronoi diagram is for the appropriate distance function, either scaled (Minkowski) or offset polygon. As before, $S$ is the input point set, but now $P$ is the fixed inner polygon.

**Lemma 11.** *The feasible region is the complement of the union of the interiors of the $n$ reflected copies of $P$ placed at points of $S$.*

**Proof.** Follows from Observation 5. □

Our first algorithm is a Voronoi-diagram approach, analogous to that of the previous section. However, it has a different feasibility region as described in Lemma 11. Note that this (polygonal) feasible region may have two different types of vertices. One type (denoted a *P-vertex*) is simply a vertex of a reflected copy of the polygon. The second vertex type (denoted an *I-vertex*) is an intersection of two copies of the reflected polygon. The following observation follows from the definitions.

**Observation 12.** *An I-vertex is equidistant from two points in $S$ according to the polygon distance function.*

Note that if we move counterclockwise around the feasible region, every traversed P-vertex is a left turn whereas I-vertices are right turns. In particular, from the point of view of the feasible region, the angle around a P-vertex that belongs to the feasible region is greater than $\pi$. This yields the next observation:

**Observation 13.** *If $e$ is the edge of a feasible region adjacent to a P-vertex, and $\ell$ is the line containing the edge $e$, then it is possible to move some distance $\varepsilon > 0$ in both directions along $\ell$ from the P-vertex without leaving the feasible region.*

Let $U$ be the union of the $n$ reflected copies of $P$ placed at the points of $S$. By Observation 12, placing $P$ at an I-vertex of the boundary of $U$ results in $P$ having at least two points of $S$ on its boundary. This means that an I-vertex of $U$ is on an edge or on a vertex of the nearest-neighbor Voronoi diagram of $S$. Furthermore, since each edge of this diagram corresponds to two points in $S$, and since the two copies of the reflected polygon associated with those points intersect in at most two points [19], each Voronoi edge can be associated with at most two I-vertices. Since the Voronoi diagram (in its compact representation) has $O(n)$ edges (each of which may be a polyline of complexity $m$ in the full representation) and $O(n)$ vertices, $U$ can have at most $O(n)$ I-vertices. (This can also be inferred from [19]. In contrast, $n$ polygons can certainly intersect in $\Theta(n^2)$ points, but only

$O(n)$ of these points may be I-vertices of the boundary of $U$. The rest are interior to $U$.) There may also be at most $O(nm)$ P-vertices. Therefore, the complexity of the boundary of $U$ is $O(nm)$. The polygon $U$ can be computed in $O(nm(\log n + \log m))$ time (see Section 3.2).

We summarize with the following:

**Theorem 14.** *The union $U$ of $n$ reflected copies of $P$ has $O(n)$ I-vertices and $O(nm)$ P-vertices, and the complexity of its boundary is $O(nm)$. It can be computed in $O(nm(\log n + \log m))$ time.*

Given an edge $e$ of the furthest-neighbor Voronoi diagram of $S$, we refer to the two points of $S$ equidistant from $e$ as the *generators* of $e$.

**Theorem 15.** *The center of the smallest enclosing polygon is in the feasible region on one of the following*:

1. *A vertex of the furthest-neighbor Voronoi diagram*;
2. *A point on an edge of the furthest-neighbor Voronoi diagram provided it is the median of the bisector of its generators* (*see Observation* 1);
3. *The intersection point of an edge $e$ of the furthest-neighbor Voronoi diagram and the boundary of the feasible region that is closest to the median of the bisector of the generators of $e$*; *or*
4. *An I-vertex of the feasible region* (*see Observation* 2).

**Proof.** Assume that the center of the minimum polygon annulus lies in the feasible region. We will show that if the center is not in one of the four places listed in the theorem, then the outer polygon can always be shrunk and thus contradicts our assumption. The center $c$ lies in the feasible region and must lie either in the interior of a Voronoi cell, on an edge of the furthest-neighbor Voronoi diagram, or on a vertex of this diagram. If $c$ lies on a vertex of the diagram, we are in case 1. If $c$ lies on an I-vertex of the feasible region, we are in case 4. Suppose $c$ is on an edge $e$ of the Voronoi diagram and in the interior of the feasible region. If $c$ is on the median of the bisector of the generators of $e$, then we are in case 2. Otherwise, moving $c$ toward the median reduces the size of the outer polygon, which is a contradiction.

Suppose $c$ is on an edge $e$ of the Voronoi diagram and on an edge of the feasible region. If $c$ is on the median of the bisector of the generators of $e$, then we are in case 2. If we can move $c$ toward the median while remaining in the feasible region, then we reduce the size of the outer polygon, which is a contradiction. If we cannot move $c$ toward the median while remaining in the feasible region, then we are in case 3. Suppose that the center $c$ lies in the interior of the furthest-neighbor Voronoi cell of a point $q \in S$ and in the interior of the feasible region. This implies that $q$ is on the boundary of the outer polygon. Furthermore, no other point of $S$ lies on the boundary of the outer polygon; therefore, it is possible to reduce the size of the outer polygon by moving $c$ toward $q$, which is a contradiction.

Finally, suppose that $c$ lies in the interior of the furthest-neighbor Voronoi cell of a point $q \in S$ and is on the boundary of the feasible region. If $c$ is on an I-vertex, we are in case 4. If

$c$ is on an edge, then by Observation 2 there is a direction along which we can move $c$ while reducing the size of the outer polygon, thereby contradicting our assumption. Similarly, if $c$ is on a P-vertex, then by Observation 13 there is also a direction along which we can move $c$ while reducing the size of the outer polygon. This completes the proof. □

**Corollary 16.** *The optimal placement of the annulus region, when its inner boundary is fixed, either has two contact points between the set $S$ and the outer boundary of the annulus region, or has three contact points with both boundaries of the region, at least one of which is in contact with the outer boundary* (*the minimized outer polygon*).

### *5.2. The algorithm*

Theorem 15 implies a natural approach to computing the minimum polygon annulus. First, compute all the possible locations for the center as listed in the theorem. Second, for each location, compute the size of the annulus. Output the smallest of these annuli.

First, we need a way of deciding whether a point $x$ is in the feasible region. To do this we compute a compact representation of the furthest-neighbor Voronoi diagram of $S$ based on a reflection of a convex polygon $P$ and the appropriate (scaled or offset) distance function. The computation of the Voronoi diagram requires $O(n(\log n + \log m) + m)$ time (for scaling) or $O(n(\log n + \log^2 m) + m)$ time (for offsetting). Alternatively, we can compute explicit representation of the Voronoi diagram in $O(nm(\log n + \log m))$ time. We preprocess the diagram for planar point location. Once we know the closest point of $S$ to $x$, we can determine whether it is in the feasible region or not. Each such query requires $O(\log n + \log m)$ time. There are $O(n)$ vertices in the furthest-neighbor Voronoi diagram. Containment in the feasible region can be checked in $O(n(\log n + \log m))$ time: there are $O(n)$ medians in the furthest-neighbor Voronoi diagram, each can be verified in $O(\log n + \log m)$ time. There are $O(n)$ vertices on the boundary of the feasible region (see Theorem 14). All of these vertices can be verified in $O(n(\log n + \log m))$ time.

To compute the intersection point of an edge $e$ of the furthest-neighbor Voronoi diagram and the boundary of the feasible region that is closest to the median of the bisector of the generators of $e$, we note that each edge of the diagram is a polygonal chain of at most $m$ segments. If the median is on $e$ and is feasible, then no other candidate on $e$ is smaller (distance-wise). Therefore, we need only consider the edges where the median is not feasible. In this case, we direct the segments of the edge toward the median. For each segment we need only the first intersection with the feasible region. This can be viewed as a ray shooting query. For each directed segment $\overrightarrow{st}$, we seek its intersection point with $U$ that is closest to $s$. Preprocessing $U$ for ray shooting queries is too costly. Instead, we perform two plane sweeps to compute the intersections between $U$ and the directed segments, one for the segments directed to the left and one for the segments directed to the right. After the first intersection for a given segment is found, we remove it from the event queue. Therefore, each segment is processed at most twice, once when it is placed in the queue and once for its first intersection. Since there are $O(nm)$ segments and the boundary of $U$ has $O(nm)$ segments, each of the two sweeps takes $O(nm(\log n + \log m))$ time. All of the candidates are generated and verified in $O(nm(\log n + \log m))$ time. We conclude with the following:

**Theorem 17.** *The minimum polygon annulus with a fixed inner polygon can be computed in* $O(nm(\log n + \log m))$ *time.*

## 6. Smallest enclosing polygon

In this section we solve Problem 3 which is a special case of Problem 2 in which the inner "radius" of the annulus is set to 0. Namely, we seek the translation of a minimum offset or scaled version of an input polygon $P$, so that it fully covers a given set $S$ of $n$ points. The problem could be solved by searching the vertices and edges of the furthest-neighbor Voronoi diagram of the respective polygon distance function. However we provide two algorithms for this problem that are more efficient than computing the entire furthest-neighbor Voronoi diagram.

### *6.1. Shrinking the feasible region*

Our first approach makes use of the results of Section 3.1. We present the algorithm and then explain both its correctness and running time.

1. Compute an offset (or scaled) version of $P$ (denoted as $P^* = O_{P,\delta}$), for some $\delta > 0$ large enough so that there exists a placement of $P^*$ containing $S$.
2. Compute the intersection $J$ of the $n$ reflected copies of $P^*$ translated to the points of $S$.
3. Shrink $J$ (by reducing $\delta$) until it becomes a single point $c$. Simultaneously shrinking $P^*$ by the same amount and translating it to $c$ produces the smallest containing polygon.

The first step is straightforward. We find in $O(n)$ time the axis-parallel bounding-box of $S$. Let $C$ be the maximum of the height and width of $B$. Then the diameter of the set is at most $C\sqrt{2}$. If we offset $P$ outward by $C\sqrt{2}/2$ in $O(m)$ time we are guaranteed to have an offset polygon $P^*$ that contains a circle of diameter $C\sqrt{2}$ and thus is large enough to contain $S$.

By Observation 4, it is guaranteed that the region $J$ (computed in the second step), that contains all placements of $P^*$ that fully cover $S$, is nonempty. Furthermore, the region $J$ is convex, with edges parallel to the original edges of $P^*$, and thus the complexity of $J$ is $O(m)$.

The crucial observation is that by reducing $\delta$ (during the third step), the above region shrinks too until it becomes a single point defining the placement and size of the *smallest* copy of $P$ that contains all the points of $S$. This observation yields the algorithm. In the second step we compute the intersection $J$ of the $n$ reflections of $P^*$ (the region of all placements of $P^*$ fully covering $S$), and in the third step we decrease $\delta$ until $J$ shrinks into a point. More specifically, we use the medial axis center (or the equivalent scaling center) of $J$ to determine the point (or segment) to which the polygon shrinks.

A solution to the second step was described in Section 3.1. The intersection polygon $J$ is computed in $O(nm)$ or $O(n \log h + m)$ time. The third step depends on whether the

polygon is offset or scaled. For the offset operation, the point to which $J$ shrinks is the center of its medial axis. (This is easily seen when we model the effect of reducing $\delta$ on $J$: the edges of $J$ are portions of edges of $P$ translated to the points of $S$.) This point can be found in $O(m)$ time by using the method of Aggarwal et al. [2]. For scaled polygons we need to slightly modify the method of [2]. The method observes that the medial axis of a convex polygon is actually the lower envelope of three-dimensional planes cutting through the edges of $P$ at fixed angles to the plane $z = 0$ that contains $P$. For the scaling operation, all we need to do is to adjust the *slope* of every plane. It is a function of three points: the origin and the two endpoints of the respective edge. Namely, the slopes are no longer fixed but are proportional to the "speeds" by which the edges move. We keep track of the original copy of $P$ to which each edge in $J$ belongs, so we can compute all these angles and solve the problem again in $O(m)$ time. The total time complexity of this algorithm is thus dominated by the second step.

In summary:

**Theorem 18.** *The smallest enclosing* (*scaled or offset*) *polygon problem can be solved in either* $O(nm)$ *or* $O(n \log h + m)$ *time.*

### *6.2. A randomized incremental approach*

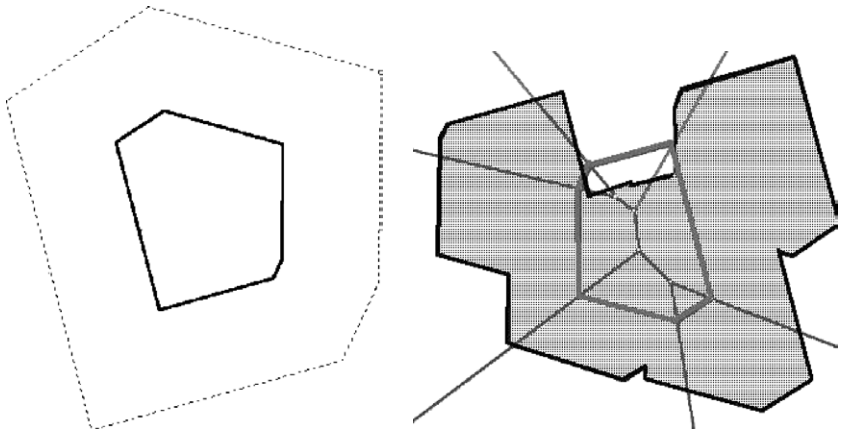
Problem 3 can also be solved by a randomized incremental approach, which is a modified version of that described in [9, §4.7] for finding the smallest enclosing circle. We start with finding the smallest enclosing polygon $P_3$ of three points $q_1, q_2, q_3 \in S$. We add point $q_i$ at the $i$th step (for $4 \leqslant i \leqslant n$). If $q_i$ is contained in $P_{i-1}$, then $P_i = P_{i-1}$. If not, we compute $P_i$ with the knowledge that the point $q_i$ must be one of the constraining points (e.g., $q_i$ lies on the boundary of $P_i$). The reader is referred to [9] for details. The analysis of the expected running time is the same as for circles except for one detail: computing the smallest (scaled or offset) polygon containing 3 points requires $O(\log m)$ time (for scaling) [22] or $O(\log^2 m)$ time (for offsetting) [6], rather than $O(1)$ time.

**Theorem 19.** *The smallest enclosing polygon problem can be solved in expected time* $O(n \log m + m)$ (*for scaling*) *or* $O(n \log^2 m + m)$ (*for offsetting*).

## 7. A new solution to Problem 2

### *7.1. The algorithm*

We now take our technique from Section 6.1 and show how it can be used to provide a new solution to Problem 2. The idea is to find some initial $\delta$ large enough to guarantee a containing annulus translation, and then to shrink it down as with our solution to the smallest enclosing polygon problem, except we are now constrained within some feasibility region that defines where the inner polygon remains empty of points. As before, for some large enough $\delta > 0$, we compute the intersection of the $n$ reflected copies of $O_{P,\delta}$ translated to the points of $S$. We call this intersection $\mathcal{I}_\delta$ (omitting the dependency on $P$



(a) (b)

Fig. 3. The union and intersection of reflections of a polygon $P$ translated to all points of a set $S$. (a) $P$ and $O_{P,\delta}$. (b) $U$ and $\mathcal{I}_\delta$.

and $S$), or just $\mathcal{I}$ when $\delta$ is clear from the context. We also compute the union of the $n$ reflected copies of $P$ translated to the points of $S$, which we denote by $U$.

Fig. 3(a) shows a sample polygon $P$ (solid) and an outer offset $O_{P,\delta}$ (dashed). Fig. 3(b) shows (as a shaded simple polygon with a solid boundary) the union $U$ of several copies of a reflection of $P$, and also (as a lighter grey polygon with its medial axis) the intersection $\mathcal{I}$ of several reflected copies of the larger offset. (Note that the union polygon $U$ is not necessarily a single polygon, but may be a collection of polygons with holes.)

If $q$ is any point that is contained in $\mathcal{I}$ but is not properly contained in $U$, then a translation of the original $P$ and $O_{P,\delta}$ to $q$ gives a containing placement of the annulus region for the set $S$. However it is not yet a solution to Problem 2 because $\delta$ is not minimized. What we want to do is to shrink $\delta$ down to the smallest value such that $\mathcal{I}$ has a nonempty intersection with the boundary or exterior of $U$.

This leads to the following algorithm:

1. Compute an outer offset $O_{P,\delta}$ of $P$ for some $\delta > 0$ large enough so that there exists a placement of the annulus region between $P$ and $O_{P,\delta}$ containing $S$.
2. Compute the intersection $\mathcal{I}_\delta$ of the $n$ reflected copies of $O_{P,\delta}$ translated to the points of $S$.
3. Compute the union $U$ of the $n$ reflected copies of $P$ translated to the points of $S$.
4. Find $\delta^*$, the minimum value of $\delta$ such that $\mathcal{I}_{\delta^*}$ contains a point exterior to or on the boundary of $U$.

Before giving more precise details about the algorithm, we make a few further observations about steps 1 and 4. In the first step we need to compute a value of $\delta$ large enough such that $\mathcal{I}_\delta$ is not empty, that is, a value that guarantees an annulus region large enough to contain $S$. (We can't shrink the annulus down if it is not big enough to start with.) To this end we note that there is a semi-circle of radius $\delta$ that lies in the annulus region between $P$

and $O_{P,\delta}$, and is centered in $v$, for every vertex $v$ of $P$. Let $w$ be the width of some axis-parallel bounding square that contains all of $S$. Then, for $\delta = w\sqrt{5}/2$ there is a semi-circle centered at the leftmost vertex of $P$ that lies in the annulus region between $P$ and $O_{P,\delta}$ and which is large enough to contain the bounding square around $S$.[7] Now we consider the final step. If we offset $\mathcal{I}_\delta$ inward by some amount, say $\alpha$, the resulting polygon is simply $\mathcal{I}_{\delta-\alpha}$, the intersection polygon that would have resulted if the original outer offset had been $\delta - \alpha$ instead of $\delta$. So in order to compute the minimal outer offset $\delta^*$, we really need only compute the value of $\alpha$ that determines how far inward the polygon $\mathcal{I}_\delta$ can be offset.

This leads to a further observation. Equivalently to offsetting $\mathcal{I}$ inward until it no longer contains a point that is not properly contained in $U$, we could compute $\mathrm{MA}(\mathcal{I})$ and consider offsetting $\mathcal{I}$ outward from its center until it contacts the first point that is not properly contained in $U$. (This approach is similar to that taken in Section 6.1.) Thus we have:

**Lemma 20.** *Let the center $c$ of $\mathrm{MA}(\mathcal{I})$ be inside $U$. Consider an expanding offset of $\mathcal{I}$ that begins at $c$ and grows outward. Then there is some point $x$ that is a first point on the boundary of $U$ hit by this expanding offset, and $x$ is either a reflex vertex of $U$ or is on the intersection of $\mathrm{MA}(\mathcal{I})$ and the boundary of $U$.*

**Proof.** We prove the claim by contradiction. Let $x$ be a first point on the boundary of $U$ that is hit by the expanding offset of $\mathcal{I}$. Suppose that $x$ is neither a reflex vertex of $U$, nor a point on $\mathrm{MA}(\mathcal{I})$. It follows that $x$ falls on some edge $e_i$ of the expanding $\mathcal{I}$ but not on a vertex of $\mathcal{I}$ (since the vertices of $\mathcal{I}$ move outward along $\mathrm{MA}(\mathcal{I})$). Suppose $x$ is on a vertex of $U$. By our assumption, it is not a reflex vertex, and so it is a convex vertex with respect to the interior of $U$. Thus at least one edge of $U$ adjacent to $x$ is interior to $\mathcal{I}$ at the moment the expanding $e_i$ contacts $x$, but this would mean that $e_i$ intersected that edge before intersecting $x$, which is a contradiction of our assumption that $x$ is an initial contact.

Suppose instead, then, that $x$ is on an edge $e_u$ of $U$, but is not a vertex of $U$. If $e_u$ is not parallel to $e_i$, then one direction along $e_u$ is closer to the inside of $\mathcal{I}$ and therefore $e_i$ will intersect $e_u$ before it reaches $x$, which is a contradiction. However, if $e_i$ and $e_u$ are parallel, then the initial point of contact is a segment one of whose endpoints is either a vertex of $e_i$ (and thus on $\mathrm{MA}(\mathcal{I})$ which is a contradiction) or is a convex vertex of $e_u$, which we assumed was not the case. □

We now present a more detailed version of step 4 of the last algorithm, enhancing the details of the final step:

4. (a) Compute $\mathrm{MA}(\mathcal{I})$, and let $c$ be its center.
   (b) Determine whether $c$ is properly contained in $U$. If it is not, then we are done. We let $\alpha$ be the amount by which we offset $\mathcal{I}$ inward until it degenerates to the point $c$. Then our $\delta - \alpha$ is the width of the smallest annulus, and $c$ is the translation of the annulus bounded by $P$ and $O_{P,\delta-\alpha}$ that contains $S$.

[7] $\sqrt{5}/2$ is the radius of a circle circumscribing a unit square, where the center of the circle is located at the midpoint of one of the edges of the square.

(c) If $c$ is properly contained in $U$, then we compute (using Lemma 20) the smallest inner offset $\alpha$ of $\mathcal{I}$ that contains a point $x$ not properly contained in $U$. Our optimal annulus region is $\delta - \alpha$ and its containing translation is given by $x$.

### 7.2. Analysis

Step 1 requires $O(n + m)$ time: $O(n)$ time is required to compute a bounding square of $S$ and $O(m)$ time to offset $P$ by this much. In step 2 we compute the intersection of $n$ copies of a convex polygon in $O(n \log n + m)$ time (or in $O(n \log h + m)$ time, where $h$ is the size of the convex hull of $S$).

In step 3 the union of $n$ copies of a convex polygon has complexity $O(nm)$ and an explicit version of it can be computed in $O(nm(\log n + \log m))$ time (Section 3.2). We can compute $MA(\mathcal{I})$ (in step 4) in $O(m)$ time by using the technique of [2].

The last two parts of step 4 are the most complex. We can perform the point-location query of $c$ in $O(\log n + \log m)$ time. We then use ray-shooting for each of the $m$ edges of $MA(\mathcal{I})$ to determine where they intersect $U$. Conversely, we test each of the $n$ reflex vertices of $U$ to determine in which region of $MA(\mathcal{I})$ it falls and then compute the offset at which the edge sweeps through it.

In the next section we provide one further enhancement of the algorithm and summarize its running time.

### 7.3. Using a compact representation of the union

The running time of the algorithm can be reduced by almost a linear factor (in the case when $m$ and $n$ are both large) by using a compact representation of $U$: the union of the $n$ copies of the reflection of $P$. Note that $U$ has complexity $O(nm)$, but only $O(n)$ of those vertices are reflex vertices representing the intersections of the boundaries of two reflected copies of $P$, since the copies of $P$ form a family of $n$ pseudo-disks [19]. Furthermore, all the reflex vertices of $U$ are of these $O(n)$ intersection-type vertices. The rest of the vertices are from some copy of the reflection of $P$.

We want to compute a representation of $U$ that explicitly stores only these intersection vertices. As noted in Section 3.2, the portions of $U$ in between these intersection vertices are just parts of chains of a copy of a reflection of $P$ and are stored implicitly with two points that specify what portion of a chain of which copy. This compact structure $U^*$ can be computed in $O(n \log n(\log n + \log m) + m)$ time by using a divide-and-conquer strategy (Section 3.2). The reflex vertices needed in step 4(c) (see Lemma 20) are explicitly stored in $U^*$. It is only slightly more complex to compute the intersection of $MA(\mathcal{I})$ with $U$. We perform a $O(\log n)$-time ray-shooting query on $U^*$ to determine which portion of a polygon the ray from $MA(\mathcal{I})$ passes through, and then a second $O(\log m)$-time ray-shooting query on that particular portion of a polygon. (Note that these are not nested steps, since we don't need to perform the second ray-shooting query until we know which region of $U^*$ we are searching.)

Thus, we have shown the following:

**Theorem 21.** *Given a convex $m$-gon $P$ and a set $S$ of $n$ points in the plane, we can determine the translation for the minimum outer offset of $P$ that contains all the points of $S$ in* $O(n \log n(\log n + \log m) + m)$ *time.*

As in Section 6.1, this technique applies to scaled as well as offset annuli, but with scaled polygons we replace the medial axis with the modified axis (as explained there) based on each edge moving at a different speed.

## 8. Conclusion and open problems

In this paper we give efficient algorithms for Problems 1 and 2, finding the smallest constrained annulus containing a set $S$ of $n$ points, where the annulus is defined by a convex $m$-gon $P$ and the offset operation, and either the outer or inner boundary of the annulus is fixed. These algorithms are simpler than previous approaches and asymptotically faster.

We conclude by mentioning a few open problems:

**Problem 5.** Set a theoretical lower bound on the asymptotic running time required to solve Problem 2.

**Problem 6.** Give efficient solutions for the annulus placement problems when the annulus is defined by a simple polygon (not necessarily convex).

**Problem 7.** Give efficient solutions for Problem 2 for polyhedra in 3-space.

## Acknowledgements

The authors wish to thank Amy Briggs, Iuliana Marinov, and Jelena Ignjatovic for work on the implementations reported in Appendix A. Many thanks are also due to an anonymous referee for meticulously reading this manuscript and for many suggestions to improve it.

## Appendix A. Experimental results

As an experimental project, we implemented five of the six algorithms proposed in Section 3.1: Brute Force 1 (BF1), Brute Force 2 (BF2), Brute Force 1 with convex-hull preprocessing (BF1-CH), Brute Force 2 using convex hulls and rotating calipers to find extremal points (BF2-CH-RC), and Brute Force 2 using binary search to find extremal points (BF2-CH-BS). The algorithms were implemented in Java and tested on numerous types of polygons and point sets.

First, we implemented two procedures for computing the intersection of two arbitrary convex polygons (used in some of the methods). Our first procedure computes the intersection by merging the list of halfplanes defining the convex polygons into a single sorted

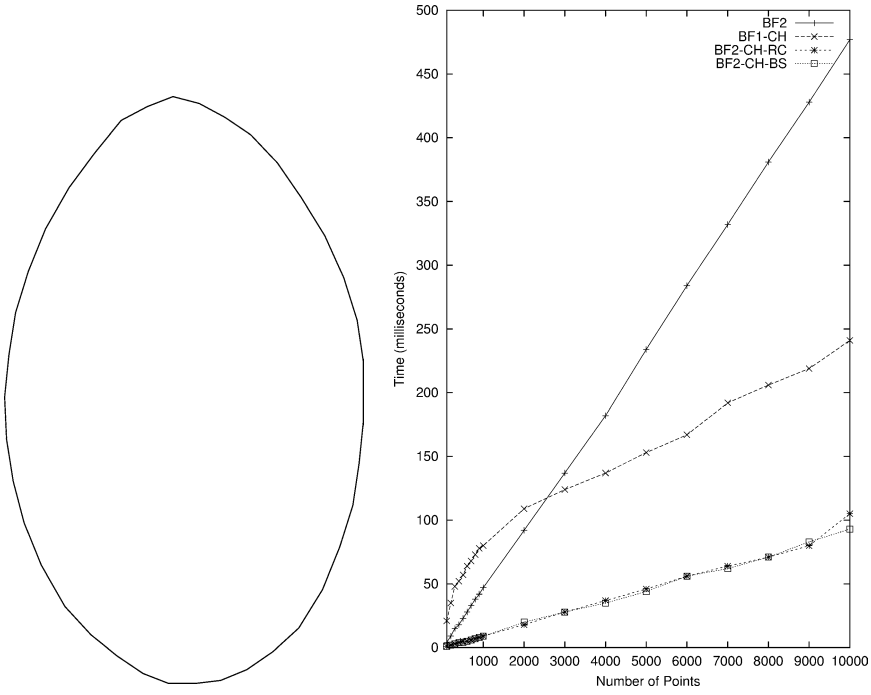



Fig. A.1. A graphical representation of the data in Table A.1(b). (a) The polygon. (b) Plot of running times.

list, and then using a Graham-Yao scan-like approach for computing the convex hull as the intersection of all those halfplanes. This approach required 190 lines of Java code. We also integrated O'Rourke's code for computing the convex hull.[8] O'Rourke's software consisted of 260 lines of Java code and initial tests showed that it was slightly faster for arbitrary convex polygons, but it crashed with polygons that had parallel edges, which is always the case for our intersections since we intersect translated copies of the same polygon. For computing the convex hull we implemented a recursive version of `QuickHull`.[9] Finally, the implementations of the BF1, BF2, BF1-CH, BF2-CH-RC, and BF2-CH-BS methods required 12, 15, 26, 45, and 23 lines of JAVA code, respectively. The first two counts do not include calls to the pairwise-intersection procedure; the last three counts exclude the code for computing the convex hull.

We used three types of polygons: (1.a) regular $m$-gons with $3 \leqslant m \leqslant 12$; (1.b) random $m$-gons, for $3 \leqslant m \leqslant 12$, where all vertices were on a circle; and (1.c) convex polygons taken from MRI contour data.[10] Here the most complex polygon contained 38 vertices. In

[8] This code is a Java implementation of the algorithm presented in [28]. It was taken from the author's web site http://cs.smith.edu/~orourke/books/CompGeom/CompGeom.html.

[9] The expected running time of our implementation of this algorithm was $O(n \log n)$ instead of possibly $O(n \log h)$, where $n$ is the number of points and $h$ is the number of hull points. Therefore, the convex-hull implementations can be made even better for large values of $n$.

Table A.1
Performance of the five polygon-intersection algorithms (times are in milliseconds)

(a) A regular 12-gon (type 1.a) with points inside it (type 2.a)

| # of points | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1,000 |
|---|---|---|---|---|---|---|---|---|---|---|
| BF1 | 30 | 46 | 72 | 89 | 101 | 122 | 144 | 166 | 196 | 215 |
| BF2 | 2 | 3 | 4 | 5 | 6 | 8 | 9 | 11 | 12 | 13 |
| BF1-CH | 5 | 5 | 6 | 6 | 6 | 7 | 8 | 8 | 9 | 9 |
| BF2-CH-RC | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| BF2-CH-BS | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 |

| # of points | 2,000 | 3,000 | 4,000 | 5,000 | 6,000 | 7,000 | 8,000 | 9,000 | 10,000 |
|---|---|---|---|---|---|---|---|---|---|
| BF1 | 422 | 568 | 755 | 905 | 1,034 | 1,267 | 2,357 | 3,052 | 3,427 |
| BF2 | 25 | 38 | 52 | 66 | 79 | 91 | 103 | 116 | 129 |
| BF1-CH | 12 | 17 | 21 | 25 | 28 | 32 | 40 | 45 | 50 |
| BF2-CH-RC | 7 | 10 | 13 | 15 | 18 | 22 | 25 | 28 | 32 |
| BF2-CH-RC | 6 | 10 | 13 | 16 | 19 | 22 | 25 | 28 | 31 |

(b) A random 7-gon (type 1.b) with points around it (type 2.b)

| # of points | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1,000 |
|---|---|---|---|---|---|---|---|---|---|---|
| BF1 | 18 | 36 | 59 | 79 | 99 | 124 | 148 | 166 | 186 | 205 |
| BF2 | 1 | 2 | 3 | 4 | 4 | 5 | 6 | 7 | 7 | 9 |
| BF1-CH | 5 | 6 | 7 | 8 | 9 | 11 | 11 | 12 | 12 | 13 |
| BF2-CH-RC | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 |
| BF2-CH-BS | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 5 |

| # of points | 2,000 | 3,000 | 4,000 | 5,000 | 6,000 | 7,000 | 8,000 | 9,000 | 10,000 |
|---|---|---|---|---|---|---|---|---|---|
| BF1 | 417 | 623 | 829 | 1,032 | 1,240 | 1,380 | 1,564 | 1,746 | 1,951 |
| BF2 | 16 | 25 | 34 | 42 | 51 | 56 | 64 | 71 | 79 |
| BF1-CH | 20 | 27 | 32 | 38 | 43 | 46 | 51 | 57 | 63 |
| BF2-CH-RC | 10 | 15 | 19 | 25 | 30 | 34 | 40 | 44 | 50 |
| BF2-CH-BS | 9 | 15 | 20 | 24 | 29 | 31 | 36 | 42 | 54 |

(c) A 38-gon (type 1.c) with points around it (type 2.b)

| # of points | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1,000 |
|---|---|---|---|---|---|---|---|---|---|---|
| BF1 | 52 | 116 | 184 | 231 | 288 | 348 | 410 | 472 | 535 | 587 |
| BF2 | 4 | 9 | 15 | 18 | 23 | 28 | 33 | 38 | 42 | 47 |
| BF1-CH | 21 | 35 | 48 | 52 | 57 | 64 | 68 | 73 | 78 | 80 |
| BF2-CH-RC | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 |
| BF2-CH-BS | 1 | 2 | 3 | 4 | 4 | 5 | 6 | 7 | 8 | 9 |

| # of points | 2,000 | 3,000 | 4,000 | 5,000 | 6,000 | 7,000 | 8,000 | 9,000 | 10,000 |
|---|---|---|---|---|---|---|---|---|---|
| BF1 | 1,186 | 1,725 | 2,252 | 2,826 | 3,389 | 4,430 | 5,125 | 5,751 | 6,377 |
| BF2 | 92 | 137 | 182 | 234 | 284 | 332 | 381 | 428 | 477 |
| BF1-CH | 109 | 124 | 137 | 153 | 167 | 192 | 206 | 219 | 241 |
| BF2-CH-RC | 18 | 28 | 37 | 46 | 56 | 64 | 71 | 80 | 105 |
| BF2-CH-BS | 20 | 28 | 35 | 44 | 56 | 62 | 71 | 83 | 93 |

addition, we generated point sets (locations of the copies of the polygon) in two ways: (2.a) points spread in the polygon interior with a uniform distribution; and (2.b) points located in an $\varepsilon$-neighborhood of the boundary of a reflected copy of it.[11] The points were first spaced equally along the polygon, then each point was offset independently from the polygon with a uniform distribution in the range $[-\varepsilon, \varepsilon]$. Naturally, the number of copies of the polygon was identical to the number of points. In all our experiments the intersection of the $n$ copies of a polygon was guaranteed to be nonempty. All the running times are averages over 100 trials, each on a different random point set. All competing algorithms were tested on the same sets of points. The size of the point sets ranged from 100 to 10,000.

The software was run on a 864 MHz Pentium III Dell computer with 256 KB cache memory and 248 MB of RAM. Table A.1 shows the results of running the five methods on three different polygons with numerous point sets. A plot of the data in Table A.1(c) is shown in Fig. A.1. The size of the 38-gon was about $250 \times 250$ units. For the experiments with this polygon the value of $\varepsilon$ was 10 units. The graph of the BF1 method was omitted because it was completely out of the scale of the other graphs. (That is, the BF1 method was much slower than all the others.) As was demonstrated in these experiment (as well as in many others), the two leading methods were BF2-CH-RC and BF2-CH-BS.

## References

[1] P.K. Agarwal, B. Aronov, M. Sharir, Exact and approximation algorithms for minimum-width cylindrical shells, Discrete Comput. Geom. 26 (2001) 307–320.
[2] A. Aggarwal, L.J. Guibas, J. Saxe, P.W. Shor, A linear-time algorithm for computing the Voronoi diagram of a convex polygon, Discrete Comput. Geom. 4 (1989) 591–604.
[3] O. Aichholzer, D. Alberts, F. Aurenhammer, B. Gärtner, A novel type of skeleton for polygons, J. Universal Comput. Sci. 1 (1995) 752–761 (an electronic journal).
[4] O. Aichholzer, F. Aurenhammer, Straight skeletons for general polygonal figures in the plane, in: Proc. 2nd Ann. Int. Computing and Combinatorics Conf., Hong Kong, in: Lecture Notes in Computer Science, vol. 1090, Springer-Verlag, Berlin, 1996, pp. 117–126, Also appeared, in: P. Engel, H. Syta (Eds.), Voronoi's Impact on Modern Science, Book 2, Inst. of Mathematics of the National Academy of Sciences of Ukraine, Kiev, 1998, pp. 7–21.
[5] G. Barequet, A.J. Briggs, M.T. Dickerson, M.T. Goodrich, Offset-polygon annulus placement problems, Comput. Geom. 11 (1998) 125–141.
[6] G. Barequet, M. Dickerson, M.T. Goodrich, Voronoi diagrams for polygon-offset distance functions, Discrete Comput. Geom. 25 (2001) 271–291.
[7] G. Barequet, M. Dickerson, P. Pau, Translating a convex polygon to contain a maximum number of points, Comput. Geom. 8 (1997) 167–179.
[8] M. de Berg, P. Bose, D. Bremner, S. Ramaswami, G. Wilfong, Computing constrained minimum-width annuli of point sets, Computer-Aided Design 30 (1998) 267–275.
[9] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer-Verlag, Berlin, 1997.
[10] P. Bose, P. Morin, Testing the quality of manufactured balls and disks, Algorithmica, in press.
[11] T.M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions, Discrete Comput. Geom. 16 (1996) 361–368.

[10] Available at ftp://ftp.cs.technion.ac.il/pub/barequet/psdb.

[11] Here we don't solve the annulus-matching problem, but only that of intersecting copies of the same convex polygon.

[12] T.M. Chan, Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus, Internat. J. Comput. Geom. Appl. 12 (2002) 67–85.
[13] L.P. Chew, R.L. Drysdale, Voronoi diagrams based on convex distance functions, Technical Report PCS-TR86-132, Dept. of Computer Science, Dartmouth College, Hanover, NH, 1986, Preliminary version appeared, in: Proc. 1st Ann. ACM Symp. on Computational Geometry, Baltimore, MD, 1985, pp. 235–244.
[14] O. Devillers, F.P. Preparata, Evaluating the cylindricity of a nominally cylindrical point set, in: Proc. 11th Ann. ACM-SIAM Symp. on Discrete Algorithms, 2000, pp. 518–527.
[15] O. Devillers, P.A. Ramos, Computing roundness is easy if the set is almost round, Internat. J. Comput. Geom. Appl. 12 (2002) 229–248.
[16] C.A. Duncan, M.T. Goodrich, E.A. Ramos, Efficient approximation and optimization algorithms for computational metrology, in: Proc. 8th Ann. ACM-SIAM Symp. on Discrete Algorithms, New Orleans, LA, 1997, pp. 121–130.
[17] L.W. Foster, Geo-metrics II: The Application of Geometric Tolerancing Techniques, Addison-Wesley, Reading, MA, 1982.
[18] L. Guibas, R. Motwani, P. Raghavan, The robot localization problem, in: Algorithmic Foundations of Robotics, A.K. Peters, 1995, pp. 269–282.
[19] K. Kedem, R. Livne, J. Pach, M. Sharir, On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles, Discrete Comput. Geom. 1 (1986) 59–71.
[20] J.L. Kelley, I. Namioka, Linear Topological Spaces, Springer-Verlag, Berlin, 1976.
[21] D. Kirkpatrick, R. Seidel, The ultimate planar convex hull algorithm, SIAM J. Comput. 15 (1986) 287–299.
[22] D. Kirkpatrick, J. Snoeyink, Tentative prune-and-search for computing fixed-points with applications to geometric computation, Fund. Inform. 22 (1995) 353–370.
[23] R. Klein, D. Wood, Voronoi diagrams based on general metrics in the plane, in: Proc. 5th Symp. on Theoretical Computer Science, in: Lecture Notes in Computer Science, vol. 294, Springer-Verlag, Berlin, 1988, pp. 281–291.
[24] V.B. Le, D.T. Lee, Out-of-roundness problem revisited, IEEE Trans. Pattern Anal. Machine Intell. 13 (1991) 217–223.
[25] M.K. Lee, A new convex-hull based approach to evaluating flatness tolerance, Computer-Aided Design 29 (1997) 861–868.
[26] M. McAllister, D. Kirkpatrick, J. Snoeyink, A compact piecewise-linear Voronoi diagram for convex sites in the plane, Discrete Comput. Geom. 15 (1996) 73–105.
[27] K. Mehlhorn, T. Shermer, C.-K. Yap, A complete roundness classification procedure, in: Proc. 13th Ann. ACM Symp. on Computational Geometry, Nice, France, 1997, pp. 129–138.
[28] J. O'Rourke, Computational Geometry in C, Second ed., Cambridge University Press, New York, 1998.
[29] A.A.G. Requicha, Mathematical meaning and computational representation of tolerance specifications, in: Proc. Int. Forum on Dimensional Tolerancing and Metrology, Dearborn, MI, 1993, pp. 61–68.
[30] E. Schömer, J. Sellen, M. Teichmann, C.-K. Yap, Smallest enclosing cylinders, Algorithmica 27 (2000) 170–186.
[31] T.C. Shermer, C.-K. Yap, Probing for near centers and relative roundness, in: Proc. ASME Workshop on Tolerancing and Metrology, Charlotte, NC, 1995.
[32] M. Smid, R. Janardan, On the width and roundness of a set of points in the plane, Internat. J. Comput. Geom. Appl. 9 (1999) 97–108.
[33] V. Srinivasan, H.B. Voelcker, Dimensional tolerancing and metrology, Center for Research and Technology Development, Research Report 27, The American Society of Mechanical Engineers, 1993.
[34] K. Swanson, D.T. Lee, V.L. Wu, An optimal algorithm for roundness determination on convex polygons, Comput. Geom. 5 (1995) 225–235.
[35] G.T. Toussaint, Solving geometric problems with the rotating calipers, in: Proc. IEEE MELECON, Athens, Greece, 1983, pp. 1–4.
[36] E. Welzl, Smallest enclosing disks (balls and ellipsoids), in: H. Maurer (Ed.), New Results and New Trends in Computer Science, in: Lecture Notes in Computer Science, vol. 555, Springer-Verlag, Berlin, 1991, pp. 359–370.