Offset-Polygon Annulus Placement Problems*

Gill Barequet[1], Amy J. Briggs[2], Matthew T. Dickerson[2], and Michael T. Goodrich[1]

[1] Center for Geometric Computing, Dept. of Computer Science, Johns Hopkins University, Baltimore, MD 21218. E-mail: [barequet|goodrich]@cs.jhu.edu
[2] Department of Mathematics and Computer Science, Middlebury College, Middlebury, VT 05753. E-mail: [briggs|dickerso]@middlebury.edu

Abstract. In this paper we address several variants of the polygon annulus placement problem: given an input polygon P and a set S of points, find an *optimal* placement of P that maximizes the number of points in S that fall in a certain annulus region defined by P and some offset distance $\delta > 0$. We address the following variants of the problem: placement of a convex polygon as well as a simple polygon; placement by translation only, or by a translation and a rotation; off-line and on-line versions of the corresponding decision problems; and decision as well as optimization versions of the problems. We present efficient algorithms in each case.

Keywords: optimal polygon placement, tolerancing, robot localization, offsetting.

1 Introduction

1.1 Background and Applications

In this paper we address several variants of the problem of placing an annulus defined by a given polygon such that it covers all (or a maximum number of) points of a given set of points. This problem is motivated by several applications. For example, in the *robot localization* problem (see, e.g., [GMR]), a robot should determine its current location in some environment map from a set of points obtained by a distance range sensor. Due to the inherent errors in range finding, the points usually do not define an exact match. Most points, however, fall within some distance $\delta > 0$ of the environment boundary. Thus the localization problem can be viewed as finding some *optimal* placement of the environment model (typically a polygon) with respect to the set of points and a distance $\delta > 0$. A second application is a pattern matching problem arising in computer vision (see, e.g., [HU]), where the input consists of a set of points taken from some image and a pattern (polygon) that one would like to locate in this image. A good match can be found by determining a placement of the polygon that maximizes the number of points within some distance $\delta > 0$ of the image points. Yet another application arises in geometric tolerancing. Chang and Yap [CY]

* Work on this paper by the first and the fourth authors has been supported in part by the U.S. ARO under Grant DAAH04-96-1-0013. Work by the third author has been supported in part by the National Science Foundation under Grant CCR-93-1714. Work by the fourth author has been supported also by NSF grant CCR-96-25289.

describe geometric tolerancing as being concerned with the specification of geometric shapes for use in manufacturing of mechanical parts, and they note that, since manufacturing processes are inherently imprecise, it is imperative that such geometric designs be accompanied by tolerance specifications. An instance of the tolerancing problem is to take a set of points representing an actual measurement of a manufactured object (using a coordinate-measuring machine, laser range-finder, or scanning electron microscope [DMSS]) and determine whether the manufactured object matches a polygon (the design) within some tolerance $\delta > 0$. This corresponds, for example, to the *tolerance zone* semantics described by Requicha [Re], Srinivasan [Sr], and Yap [Ya].

1.2 Previous Related Work

The notion of polygon annulus placement relative to a set of points appears to be new in the computational geometry literature. There are nevertheless several related problems that have been studied before, including variants directed at placing an entire polygon (not an annulus) to cover a set or subset of points (see, e.g., [ESZ, EE, BDP, DS]). These problems do not model important aspects for optimizing polygon placement (as mentioned in the applications above). Previous work directed at annulus problems, on the other hand, have dealt exclusively with circular annuli (see, e.g. [HT, LL, AST, AS, SJ, SLW, DGR]). These characterizations capture well the notion of "roundness" present in a set of points, but they do not easily extend to polygonal shape matching.

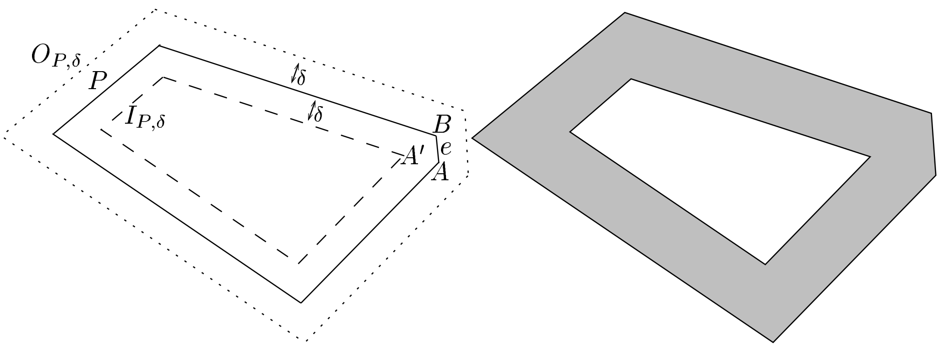
1.3 Definitions and Problems

We start with definitions for convex polygons to simplify the presentation. Extensions to simple polygons are made in Section 5.

Definition 1 (Offset Annulus) *The* δ-annulus *of a convex polygon P is the closed region defined by all points in the plane at distance at most δ from the boundary of P.*

Definition 2 (Offset Polygons) *Given a convex polygon P and a distance $\delta > 0$, the* δ-offset *polygons are defined as follows: The* inner *δ-offset polygon $I_{P,\delta}$ is the boundary portions of the δ-annulus of P that are properly contained by P. Similarly, the* outer *δ-offset polygon $O_{P,\delta}$ is the boundary portions of the δ-annulus of P outside of (i.e., properly containing) P.*

Note that $I_{P,\delta}$ is made up of edges that are parallel to edges of P (although there may be some edges of P that are not parallel to any in $I_{P,\delta}$). The offset polygon $O_{P,\delta}$, on the other hand, is made up of alternating line segments and circular arcs, and every edge of P is parallel to some edge of $O_{P,\delta}$. One can also imagine a fully *linearized* version of the outer offset polygon, where one extends each of the linear edges until they meet the extensions of neighboring linear edges. (For simplicity, we will first discuss algorithms for solving polygon-annulus problems adopting this linearized view, and we will then show how to



(a) Linearized inner and outer offset polygons (b) The δ-annulus region.

Fig. 1. Offsetting a polygon

extend these to the more-natural standard notion of a δ-offset without affecting the running times by more than a constant factor.)

Figure 1(a) shows a convex polygon P (with solid edges) and its inner and linearized outer offset polygons $I_{P,\delta}$ and $O_{P,\delta}$ (with dashed and dotted edges, respectively) for some value of δ. Note that for any convex polygon P and for any value of δ the outer offset polygon $O_{P,\delta}$ always has the same number of edges as P, but the inner offset polygon $I_{P,\delta}$ may have fewer edges. In this example the edge $e \in P$ does not have a counterpart in $I_{P,\delta}$. More specifically, the point A, edge e, and point B, all in P, collapse into a single point A' in $I_{P,\delta}$. Also, unless P is a regular polygon, the offset polygons $I_{P,\delta}$ and $O_{P,\delta}$ are *not* scaled versions of P. The δ-annulus region of a polygon is shown shaded in Figure 1(b). Note that the annulus region is defined to include the boundary edges. Although these definitions are stated for convex polygons, we show that in many cases they can easily be extended to simple polygons (see Section 5). In any case, the definition of δ-annulus regions naturally gives rise to the following problems:

- **Offset-Polygon Max Cover**: Given a set S of n points in the plane, a convex polygon P, and a distance δ, find a placement τ of P that maximizes the number of points of S contained in the δ-annulus region of $\tau(P)$. Report the placement τ and the set of contained points.
- **Offset-Polygon Containment (Decision Version):** Given a set S of n points in the plane, a convex polygon P, and a distance δ, determine if there exists a placement τ of P such that *all* n points of S are contained in the δ-annulus region of $\tau(P)$. Report such a placement τ if one exists.
- **Offset-Polygon Containment (Optimization Version):** Given a set S of n points in the plane, and a convex polygon P, find the smallest value of $\delta > 0$ such that there exists a placement τ of P with *all* n points of S being contained in the δ-annulus region of $\tau(P)$. Report such a placement τ if one exists, together with this optimal value of $\delta > 0$.

Note that we can use an algorithm for either the offset-polygon max-cover problem or for the width-optimization problem to solve the offset-polygon containment decision problem. In particular, the answer for the decision problem is "yes" if and only if for the former problem the value of k—the maximum number of points contained in the δ-annulus for P—is n, or for the latter problem the value of δ'—the minimum width of an annulus that contains all the points—is δ.

1.4 Outline and Summary of Results

Let n be the number of input points and let m be the number of edges (and vertices) of the given polygon P. In this paper we give several results for solving the offset-polygon max-cover and containment problems. We show that if we restrict the offset-polygon containment decision problem to convex polygons under translations only, then we can determine a containing placement of P, if one exists, in $O(n \log n \log m + m)$ time. Our method involves a non-trivial extension of the roundness method of Duncan et al. [DGR] to offset polygons using the polygon-offset nearest-neighbor and furthest-neighbor diagrams [BDG]. Moreover, we show how to solve the optimization version of this problem in the same time bound, by using the simplest (and most practical) version of parametric searching.

We also study the offset-polygon max-cover problem for convex polygons under translations, showing that this more-general problem can be solved in $O(n^2 \log(nm) + m)$ time and $O(n + m)$ space. Our methods involve a non-trivial extension of the techniques of Barequet et al. [BDP]. In addition, we show how to solve this problem under translations and rotations by combining this approach with extensions of the rotation-diagram techniques of Dickerson and Scharstein [DS]. The resulting time bound in this case is $O(n^3 \log(nm) + m)$ using $O(n + m)$ space in the worst case. Under some very reasonable "fatness" conditions (which we make precise in Section 5), we show that our techniques can be generalized for simple polygons under translations to result in an algorithm running in $O(n^2 m^2 \log(nm))$ time and $O(nm^2)$ space.

In addition to the off-line results discussed above, we also describe a method, based upon an interesting dynamic data structure, that solves an on-line version of the offset-polygon containment decision problem under translations. The algorithm reads points one at a time, halting and answering "no" when a placement containing all points read so far is no longer possible, or, alternatively, running to completion on n points and answering "yes." In the worst case, this on-line algorithm runs in $O(n^2 m^2 \log(nm))$ time and $O(n^2 m^2)$ space for simple polygons. For many distributions of points, however, it performs significantly better. In particular, for convex polygons our on-line algorithm runs in $O(nh \log(nm) + m)$ time and requires only $O(nh + m)$ space, where h depends on the distribution (see Section 4.3). (In the worst case $h = \Theta(n)$, but for many distributions h is substantially smaller.)

The outline of the paper is as follows. We begin in Section 2 with some important geometric properties and primitives. In Section 3 we present the algorithms for convex polygons, and in Section 4 we give our on-line solution to the offset-polygon max-cover problem. In Section 5 we extend our solutions to the offset-polygon max-cover problem to simple polygons. We conclude with Section 6.

2 Key Geometric Properties

An important step of our algorithms is the computation of the intersections between translated copies of offset polygons. For simplicity of expression, let us assume we are dealing with linearized offset polygons; we show later how to remove this restriction to deal with the more-standard definition of δ-annulus region with only a constant-factor increase in the running times of our algorithms. Let us therefore consider an upper bound on the number of intersections between translated copies of linearized offset polygons, and a description of how to compute them. It is well-known that two translated homothetic copies of the same convex polygon can intersect at most twice (where in the degenerate case an intersection may be a segment rather than a point). The following theorem states that translations of an inner and outer offset convex polygon can also intersect at most twice.

Theorem 1. *Given a polygon P, a distance δ, and a translation τ, the offset polygons $\tau(I_{P,\delta})$ and $O_{P,\delta}$ intersect at most twice, where each intersection may be a point or (in the degenerate case) a segment.*

Proof Omitted in this version of the paper. □

The technique used in this proof also provides the necessary framework for the proof of the following lemma (using the *tentative prune-and-search* technique of Kirkpatrick and Snoeyink [KS]).

Lemma 2. *The intersections between offset polygons $\tau(I_{P,\delta})$ and $O_{P,\delta}$ can be found in worst case $O(\log m)$ time, where m is the number of vertices of P.*

We compute these intersections because they correspond to placements of the annulus region such that two (or more) points of S are in contact with the boundary of the annulus region.

The following lemmas are generalizations of lemmas from [BDP, DS] that deal with intersections between two copies of the same polygon.

Lemma 3. *Let P be a convex polygon, q_1, q_2 points, and τ_1 and τ_2 the translations mapping the origin to points q_1 and q_2, respectively. For any point x, let $\tau_x = q_2 - x$ be the translation that maps x to q_2. Then both q_1 and q_2 are contained in the δ-annulus region of $\tau_x(\tau_1(P))$ if and only if x is contained in the intersection of the δ-annulus regions of $\tau_1(P)$ and $\tau_2(P)$.*

Lemma 4. *Let P be a convex polygon and S be a non-empty set of points contained in the δ-annulus of P. Then there exists a translation τ such that S is contained in the δ-annulus of $\tau(P)$ and at least one point of S is on the boundary of the annulus region.*

3 Algorithms for Convex Polygons

3.1 Offset-Polygon Containment under Translation

We first briefly describe a deterministic $O(n \log n \log m + m)$-time algorithm for solving the annulus-width optimization problem: Given a set S of n points and a convex polygon P with m vertices, find the minimum-width annulus of P that covers S. For this purpose we define the *convex polygon-offset* distance-function $\mathcal{D}_P$ that corresponds to P and compute the nearest- and furthest-site Voronoi diagrams of S with respect to $\mathcal{D}_P$ (see [BDG]). This can be performed in $O(n(\log n + \log m) + m)$ time. Next we use the method of [DGR] (where the authors minimize the width of a *circular* annulus) and consider the overlay of the two diagrams. As is well-known, the center of the minimum-width annulus that contains S is either a vertex of one of the two diagrams (possibly a vertex at infinity in the furthest-site diagram) or a point of intersection between the two diagrams. Given a specific value of δ, we place δ-annuli centered at all the points of S and observe (like in [DGR]) the overlay for determining whether the intersection of all annuli is nonempty. (The intersection contains the loci of all feasible placements of the annulus so that it covers S.) This step takes $O(n \log m)$ time. Finally, a parametric-searching algorithm is applied for optimizing (minimizing) the value of δ for which the intersection of all the annuli is nonempty. Over all, the whole procedure requires $O(n \log n \log m + m)$ time.

3.2 Offset-Polygon Max-Cover under Translation

In this section we consider offset-polygon max-cover under translation. Our algorithm extends the techniques of Barequet et al. [BDP] to allow for containment within the annulus region rather than containment by the entire polygon. The idea is to do an anchored sweep of *both* the inner and outer offset polygons around each point of S. The critical events of the sweep occur when some point of S either enters or exits the δ-annulus. The full algorithm is given in Figure 2. The correctness of this algorithm follows from Lemmas 3 and 4. There exists at least one optimal placement with a point in contact with the annulus boundary, and this placement will be found by the sweep. The only additional detail regards the processing of degenerate intersections, where the intersection between two offset polygons is a segment (along a connected portion of an edge) rather than a discrete point. In this case only one of the two endpoints of the segment corresponds to an event. If the point q_i is currently marked "in" then it is at the second endpoint of the intersection segment where it changes to "not in." Conversely for points marked "not in," it is at the first endpoint of the segment where it changes to "in." This follows from the fact that the entire segment corresponds to a translation in which both points q_i and q_j are on the boundary of the translated polygon and so points that are "in" remain so until the *end* of the segment, whereas points that are "not in" become "in" at the *start* of the segment.

We measure the complexity of our max-cover algorithm under translations as a function of two variables: m, the number of vertices of P, and n, the number

I. Preprocessing:

1. Preprocess offset polygons $I = I_{P,\delta}$ and $O = O_{P,\delta}$ for intersection computation.
2. Initialize a priority queue Q which will store points in clockwise order around the boundaries of the offset polygons I and O.

II. Iteration:

1. Set max $:= 0$. $\{\#$ of points so far$\}$
2. **FOR** each point $q_i \in S$ **DO BEGIN** $\{$Anchored sweep around $q_i\}$
3.   Let P' be I. $\{$First sweep $I\}$
4.   Set $c := 1$. $\{$Points contained$\}$
5.   **FOR** each $j \neq i$ and $q_j \in S$ **DO BEGIN** $\{$Examine nearby points$\}$
6.   Set $X := \{x | x \in \partial\tau_i(I) \cap \partial\tau_j(P')\} \bigcup \{x | x \in \partial\tau_i(O) \cap \partial\tau_j(P')\}$.
7.   **FOR** all $x \in X$ **DO**
8.   Add (x, j) to Q. $\{$Add intersections to queue$\}$
        END FOR
9.   **IF** q_j is contained in the δ-annulus of $\tau_i(P)$ **THEN**
10.   Mark q_j 'In'; Set $c := c + 1$; $\{$Mark and count points$\}$
        ELSE
11.   Mark q_j 'Not In'.
        END IF
    END FOR
12.   **WHILE** $Q \neq \emptyset$ **DO BEGIN** $\{$Sweep with intersections$\}$
13.   Delete (x, j) from front of Q. $\{$Update structures$\}$
14.   **IF** q_j is 'Not In' **THEN** $\{$See comments in text$\}$
15.   Set $c := c + 1$; Mark q_j 'In'.
        ELSE
16.   Set $c := c - 1$; Mark q_j 'Not In'.
        END IF
17.   **IF** $c >$ max **THEN**
18.   Set max $:= c$; Store translation.
        END IF
    END WHILE
19.   **REPEAT** steps 4 through 18 with $P' = O$. $\{$Now sweep $O\}$
END FOR

Fig. 2. Max-cover algorithm under translations, for convex polygons

of points in the set S. The preprocessing step requires $O(m)$ time and space for computing and storing the offset polygons $I = I_{P,\delta}$ and $O = O_{P,\delta}$. The offset polygons are stored such that later intersection tests can be performed in $O(\log m)$ time and space (see [KS] and Section 2). The steps inside the inner nested loop execute $O(n^2)$ times. Since each pair of points has two offset polygons, each of which has at most two intersections with the polygon being swept, the total size of the queue is $O(n)$ and queue operations can be performed in $O(\log n)$ time. Polygon intersections in Step 6 can be computed in $O(\log m)$ time (by Lemma 3). The total running time is therefore $O(n^2 \log(nm) + m)$ in the worst case. The algorithm requires $O(n + m)$ space. (In the full version of the paper we show how to significantly improve the running time of the algorithm in many instances by the use of bucketing.)

3.3 Offset-Polygon Max-Cover under Translation and Rotation

We now describe how the offset-polygon max-cover problem can be solved for convex polygons when we allow for translations and rotations. To solve this problem we extend the results of Dickerson and Scharstein [DS] and make use of their *rotation diagram* technique. We refer the reader to [DS] for details on this method; here we describe only the necessary modifications in the approach and in the complexity analysis. This method creates a rotation diagram R_{q_i} for each point q_i. The diagram R_{q_i} is a description of the configuration space of all placements of the polygon P that keep the boundary of P in contact with q_i. The horizontal axis of this diagram represents the angle of rotation (from 0 to 2π). The vertical axis represents the arclength along ∂P (from 0 to the circumference of P). For each other point q_j, the rotation diagram for q_i includes the region of all such placements that contain q_j. It is shown in [DS] that this containing region for q_j can be decomposed into $O(m^2)$ subregions of constant complexity. The left and right boundaries of these subregions are certain critical angles of rotation, where vertices of one polygon pass through edges of another. The upper and lower boundaries are shown to be sine curves. To solve the optimal placement problem, the algorithm performs a plane sweep of each rotation diagram R_{q_i} to find the region of greatest depth. This gives the optimal placement of P that is in contact with q_i. The main difference for the annulus placement problem is that we need two rotation diagrams for each point q_i: one for the inner offset polygon $I_{P,\delta}$ and one for the outer offset polygon $O_{P,\delta}$. Furthermore, each of these two rotation diagrams for q_i has regions for each $q_j \neq q_i$ that represent containment in the annulus region rather than in the entire polygon. The following lemma states that these modified rotation diagrams have the same complexities.

Lemma 5. *For convex polygons, the polygon annulus containing regions for a given point is decomposed into $O(m^2)$ subregions each of which have constant complexity: vertical left and right boundaries and a sine curve for the top and bottom boundaries.*

The proofs of [DS] suffice to show that the upper and lower boundaries are still sine curves. The $O(m^2)$ is a trivial upper bound which is attainable. There is however a constant factor increase in the complexity of the diagrams. The number of critical angles are doubled because we now count intersections of both the inner and outer polygon placed at point q_i and either the inner or outer polygon at q_j (depending on which rotation diagram we are computing). Therefore, since the number of subregions can double, the number of intersection points can increase by a factor of four. To solve the offset-polygon max-cover problem we use the same idea of the rotation diagram and perform plane sweeps of each of the $2n$ diagrams. Lemma 4 tells us that this suffices because even with a restriction to translation only there is at least one optimal placement that has a point on an inner or an outer boundary of the annulus region. Thus we can state the following theorem:

Theorem 6. *The convex offset-polygon maximum-cover problem can be solved in $O(n^3 \log(nm) + m)$ time and $O(n + m)$ space in the worst case for translation and rotation.*

3.4 True δ-Tolerancing

As mentioned earlier, our algorithms assume a linearized outer polygon boundary. For adapting the linearized versions of the offset-polygon max-cover and offset-polygon containment problems to their (standard) non-linear forms, we need only show that the framework for the offset-annulus translation variant works also for the true δ-tolerancing case. The key to this adaptation lies in the fact that for every convex polygon P, tolerance δ, and a translation τ, the number of intersections of $\tau(I_{P,\delta})$ and the true outer boundary $O_{P,\delta}$ is still at most two. The proof of this claim is almost identical to that of Theorem 1 (see Section 2). Indeed, the weak monotonicity of the curves is preserved (we do not need the curves to be piecewise-linear). Furthermore, we can still apply the prune-and-search technique, since the simplicity of the pieces of the curves is also maintained: it takes a constant amount of time to evaluate the intersection of a circular arc with a line segment or with another circular arc. Therefore we are able to apply the same algorithm (for the translation-only variant) as in Section 3.2 and obtain the same asymptotic running time and space. In the full version of the paper we explain how to extend also the translation and rotation version of the problem for the true δ-Tolerancing case.

4 An On-Line Decision of the Containment Problem

In the previous section we provided solutions to several variants of the offset-polygon max-cover and containment problems, under various rigid transformations. In this section we present an alternate “on-line” approach to offset-polygon containment decision problems for the translation-only case. As before, we assume convex polygons and deal with simple polygons in a later section. The idea of this on-line approach is that instead of being given the entire set S at once, the points are read one at a time, and for each new point we decide whether there is a placement of the annulus region of P that contains all the points *seen so far*. There are several motivations for the on-line approach. One is that for the decision problem we need not necessarily process the entire point set; if after a certain number of points there is no longer a placement containing them all then we can halt immediately and answer ‘No’ (thus offering some savings in running time over unnecessarily processing all the points). This may be particularly useful for the tolerancing problem. A second advantage is the ability to process incoming points *as they arrive* while simultaneously reading subsequent points (a form of pipelining). This is an advantage in the cases of the proposed applications where the points are not stored in a file but are read one-at-a-time by an external device. A third possible advantage is that as more points are read we can slowly *refine* the space of possible placements of P. This can be helpful for both the robot localization and geometric tolerancing problems where we might direct the input device for further measurements. Finally, the on-line approach allows for the *pruning* of the data structures providing a more efficient approach for most practical applications.

4.1 Basic Algorithm Approach

We begin with the basic ideas of the on-line approach. We want to read input points one at a time. For each point q_i we construct and store a data structure (similar to that of Algorithm 1) that maintains optimal placements of the annulus region around P in contact with q_i. We also update the data structures for the existing points q_j for $j < i$. That is, for each $j < i$ we: (1) Compute the translations that keep the annulus region of P in contact with q_i and contain q_j and add this information to the new data structure of q_i; and (2) Compute translations that keep the annulus region in contact with q_j and contain q_i and update the data structure of q_j. Remember that for each point q_j these translations are computed from the intersections of the translated offset polygons in $O(\log m)$ time by Lemma 2. However our use of data structures for the on-line algorithm differs in two ways from Algorithm 1. The first difference is that (unfortunately) we need to store several data structures simultaneously, rather than computing the optimal placement for one and then discarding it. This is because each data structure is continually being updated as new points are added. The second difference is more advantageous: since we are concerned only with the decision problem of whether there is a placement containing *all* n points, we need keep track of only those placements containing all points seen so far. Any placement that does not contain all points can be discarded. That is, we want the *intersections* of all the pairwise containing regions, where each region is given by a pair of segments (possibly empty) on the inner offset polygon and another pair of segments (also possibly empty) on the outer offset polygon. If at any point in the algorithm there are no such remaining placements, then we can halt and output 'No'.

4.2 Analysis and Details of Data Structure

How do we store the set of placements containing all points? Recall that the region of placements containing q_i and with q_j on the boundary corresponds to a pair of segments along the inner and outer boundaries of the annulus region. For each point, we store these placements in two balanced binary search trees (one for the inner polygon and one for the outer polygon) ordered clockwise around the boundary of the polygon. Unfortunately, it is possible to construct a case where the complexity of the set of placements containing all points is $\Theta(n)$. (Each new pair of segments increases the complexity of the arrangement by 2.) Thus the space required *per point* may be as high as $\Theta(n)$ for a total of $\Theta(n^2)$ space. The searches, inserts, and deletes can all be performed in $O(\log n)$ time. In particular, for each new point q_i added to the structure of point q_j, there are at most two segments to be added to both the inner and outer offset polygons. Since we want only placements containing all points, we store the *intersections* of these two new segments with all existing segments. We find the endpoints in $O(\log m)$ time and delete all regions not inside the endpoints. Deleting one segment and rebalancing the tree requires $O(\log m)$ time. The total number of insertions and deletions to each tree is $O(n)$. Hence we need a total of $O(n^2 \log n)$ time for updating all the trees and $O(n^2 \log m)$ time for computing all the intersections.

The overall complexity is thus $O(n^2 \log(nm) + m)$ time and $O(n^2 + m)$ space in the worst case (when no pruning is done). The algorithm may terminate early with a 'No' answer.

4.3 Improvement by Pruning

Both the space and time complexity of the algorithm can be improved considerably by an on-line pruning. Recall from the previous section that for each point q_i we need to store only the placements containing *all* the points: that is, the intersections of all $i-1$ intersection regions. We can discard the data structure of a point q_i when this intersection region becomes empty. This happens when there are no longer any placements containing all other points with q_i on the boundary. We define $H_1 = \{p_1\}$ and H_i (for $2 \le i \le n$) to contain all the points $x \in H_{i-1} \cup \{p_i\}$ such that there exists a placement $\tau(P)$ which contains $H_{i-1} \cup \{p_i\}$ with x on the boundary of $\tau(P)$. Let $h_i = |H_i|$. (In case $h_i = 0$ for some i the algorithm terminates with a 'No' answer.) Also, let h be the maximum value of h_i for $1 \le i \le n$. Then the total number of data structures after the ith step of the algorithm is h_i and the total at any time is $O(h)$. The total time required to update existing data structures for a new point q_{i+1} is $O(h_i \log n) = O(h \log n)$. The pruning step can be implemented efficiently. The main idea, which we explore in the full version of the paper, is marking points for deletion (and temporarily ignoring them), but using them at a later stage of the algorithm. We omit in this version of the paper the full analysis of the pruning version of the algorithm. The total running time of the algorithm is $O(nh \log(nm) + m)$ time. The algorithm requires only $O(nh + m)$ space in the worst case.

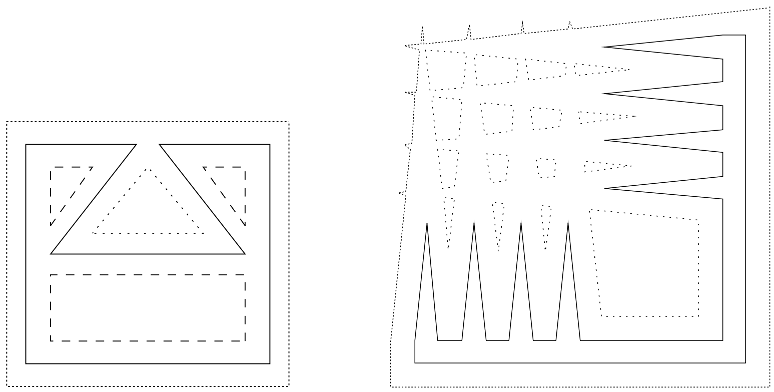
5 Simple Polygons

In this section we extend our results to the case of simple polygons.[3] We restrict the class of simple polygons to a natural set of "fat" polygons, which are more-natural candidates for offset-polygon placement problems. Specifically, we disallow polygons with narrow corridors, as specified in the following definition.

Definition 3 (δ-wide Polygons) *A* δ-wide *polygon P is a simple polygon with the property that if $p, q \in \partial P$ with $dist(p, q) \le 2\delta$ then there is a path connecting p and q along the boundary of P such that every point on the path is at most 2δ away from p or every point is at most 2δ away from q.*

This restriction is reasonable for the proposed applications since the offset polygons are meant to capture measurements that are close to the input polygon and in actual production the allowed tolerance has to be twice the minimum feature-size. It eliminates cases in which the inner and outer δ-offsets of P become disconnected or non-simply-connected. Figure 3(a) shows a simple polygon

[3] See [AAAG] for a discussion of the *straight skeleton* of a simple polygon which is closely related to the notion of the inner offset polygon. This discussion is however not in the context of the problems discussed in this paper.



(a) Non-simple inner and outer offsets (b) Intersecting offsets of narrow spikes

Fig. 3. Simple polygons with non-simple offsets

(with solid edges) whose outer offset is *perforated*: it contains a boundary (dotted densely) and a hole (dotted sparsely). The inner offset of the same polygon consists of three distinct polygons (with dashed edges). Figure 3(b) shows another simple polygon whose outer offset is perforated.

We can solve simple-polygon variants of the offset-polygon max-cover problem for δ-wide simple polygons with a slightly modified version of the algorithm given in Figure 2. Let us use $\overline{O}_{P,\delta}$ to denote the *true* (non-linear) outer δ-offset of P. Similarly, we call the inner curve formed by straight segments and circular arcs at distance δ the *true* inner δ-offset of P and denote it by $\overline{I}_{P,\delta}$. Note that $\overline{I}_{P,\delta}$ and $\overline{O}_{P,\delta}$ are each of complexity $O(m)$ for a δ-wide simple polygon P of size m. Instead of having at most two intersections between $\tau(\overline{I}_{P,\delta})$ and $\overline{O}_{P,\delta}$, we can have $\Theta(m^2)$ pairwise intersections in the worst case requiring $\Theta(m^2)$ time to compute. Each pair of points has two offset polygons, each of which has $O(m^2)$ intersections with the polygon being swept. So the size of the queue is $O(nm^2)$ and queue operations can be performed in $O(\log(nm))$ time. The overall complexity of the algorithm becomes $O(n^2m^2\log(nm))$ time and $O(nm^2)$ space. The above running times also hold for linearized versions of the offset-polygon max-cover problem if we disallow polygons with narrow features that cause the outer or inner offset polygon to intersect itself. Note that for a general simple polygon P, both linearized outer and inner boundaries, $O_{P,\delta}$ and $I_{P,\delta}$, can contain some points further than δ from ∂P. As a result, $I_{P,\delta}$ and $O_{P,\delta}$ can each be of complexity $\Theta(m^2)$ for a polygon P with m vertices. Figure 3(b) gives an illustration of this. Thus there can be $\Theta(m^4)$ intersections between $\tau(I_{P,\delta})$ and $O_{P,\delta}$. The simple polygons to which our algorithm applies must therefore be δ-wide without narrow spikes (as shown in Figure 3(b)).

The on-line algorithm can also be modified for simple polygons. If we make the assumption that the features of the polygon are such that the annulus region has $O(m)$ complexity, then in the worst case the number of intersections between two translated copies of the annulus is $O(m^2)$ and the complexity of the arrangement of containing regions for a given point is $O(nm^2)$. The on-line algorithm therefore requires $O(n^2m^2 \log(nm))$ time and $O(n^2m^2)$ space.

6 Conclusion

In this paper we provide efficient algorithms for polygon offset placement problems. We handle the convex and non-convex cases, the translation-only variant as well as the translation and rotation variant, the static and dynamic modes of input points, and decision and optimization versions of the problems. There are several possible further research directions, which include the following:

1. Minimizing δ such that the placement of the given polygon annulus contains some given value $k < n$ of them (offset-polygon partial containment).
2. Generalizing from a polygon to a collection of polygonal *chains*. (This variant often occurs in applications to robot localization.)
3. Generalizing from polygons to smooth shapes.
4. Computing *approximate* solutions to all of these problems.
5. Proving lower bounds for the problems.
6. Solving similar problems in higher dimensions.
7. Analizing the *expected* value of h in the pruning version of the on-line algorithm (Section 4.3).

Acknowledgement. We would like to thank Vincent Mirelli for several stimulating conversations related to the topics of this paper.

References

[AAAG] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner, A novel type of skeleton for polygons, *J. of Universal Computer Science* (an electronic journal), 1 (1995), 752–761

[AS] P.K. Agarwal and M. Sharir, Efficient randomized algorithms for some geometric optimization problems, *Proc. 11th Ann. ACM Symp. on Computational Geometry*, Vancouver, Canada, 1995, 326–335.

[AST] P.K. Agarwal, M. Sharir, and S. Toledo, Applications of parametric searching in geometric optimization, *J. Algorithms*, 17 (1994), 292–318.

[BDG] G. Barequet, M. Dickerson, and M.T. Goodrich, Voronoi diagrams for medial-axis distance functions, these proceedings, 1997.

[BDP] G. Barequet, M. Dickerson, and P. Pau, Translating a convex polygon to contain a maximum number of points, *Proc. 7th Canadian Conf. on Computational Geometry*, Québec City, Québec, Canada, 1995, 61–66; full version to appear in: *Computational Geometry: Theory and Applications*.

[Ch] B. Chazelle, The polygon placement problem, *Advances in Computing Research: volume 1* (F. Preparata, ed.), JAI Press, 1983, 1–34.

[CK] I.J. Cox and J.B. Kruskal, Determining the 2- or 3-dimensional similarity transformation between a point set and a model made of lines and arcs, *Proc. 28th Conf. on Decision and Control*, 1989, 1167–1172.

[CY] E.-C. Chang and C.-K. Yap, Issues in the Metrology of Geometric Tolerancing, Courant Institute of Mathematical Sciences, New York University, Unpublished manuscript.

[DGR] C.A. Duncan, M.T. Goodrich, and E.A. Ramos, Efficient approximation and optimization algorithms for computational metrology, *Proc. 8th Ann. ACM-SIAM Symp. on Discrete Algorithms*, New Orleans, LA, 1997, to appear.

[DS] M. Dickerson and D. Scharstein, Optimal placement of convex polygons to maximize point containment, *Proc. 7th Ann. ACM-SIAM Symp. on Discrete Algorithms*, Atlanta, GA, 1996, 114–121.

[DMSS] W.P. Dong, E. Mainsah, P.F. Sullivan, and K.F. Stout, Instruments and Measurement Techniques of 3-Dimensional Surface Topography, *Three-Dimensional Surface Topography: Measurement, Interpretation and Applications* (K.F. Stout, Ed.), Penton Press, Bristol, PA, 1994.

[EE] D. Eppstein and J. Erickson, Iterated nearest neighbors and finding minimal polytopes, *Discrete & Computational Geometry*, 11 (1994) 321–350.

[ESZ] A. Efrat, M. Sharir, and A. Ziv, Computing the smallest k-enclosing circle and related problems, *Computational Geometry: Theory and Applications*, 4 (1994), 119–136.

[GMR] L. Guibas, R. Motwani, and P. Raghavan, The robot localization problem, in: *Algorithmic Foundations of Robotics*, A K Peters, Ltd., 1995, 269–282.

[HT] M.E. Houle and G.T. Toussaint, Computing the width of a set, *Proc. 1st Ann. ACM Symp. on Computational Geometry*, 1985, 1–7.

[HU] D.P. Huttenlocher and S. Ullman, Recognizing solid objects by alignment with an image, *Int. J. of Computer Vision*, 5 (1990), 195–212.

[KS] D. Kirkpatrick and J. Snoeyink, Tentative prune-and-search for computing fixed-points with applications to geometric computation, *Fundamental Informaticæ*, 22 (1995), 353–370.

[LL] V.B. Le and D.T. Lee, Out-of-roundness problem revisited, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13 (1991), 217–223.

[LS] H.P. Lenhof and M. Smid, Sequential and parallel algorithms for the k closest pairs problem, *Int. J. Computational Geometry and Applications*, 5 (1995), 273–288.

[Re] A.A.G. Requicha, Mathematical meaning and computational representation of tolerance specifications, *Proc. Int. Forum on Dimensional Tolerancing and Metrology*, 1993, 61–68.

[SJ] M. Smid and R. Janardan, On the width and roundness of a set of points in the plane, *Proc. 7th Canadian Conf. on Computational Geometry*, Québec City, Québec, Canada, 1995, 193–198.

[Sr] V. Srinivasan, Role of sweeps in tolerance semantics, *Proc. Int. Forum on Dimensional Tolerancing and Metrology*, 1993, 69–78.

[SLW] K. Swanson, D.T. Lee, and V.L. Wu, An optimal algorithm for roundness determination on convex polygons, *Computational Geometry: Theory and Applications*, 5 (1995), 225–235.

[Ya] C.-K. Yap, Exact computational geometry and tolerancing metrology, in: *Snapshots of Computational and Discrete Geometry*, Vol. 3, Technical Report SOCS-94.50 (D. Avis and J. Bose, Eds.), McGill School of Computer Science, 1995.