An Efficient Proximity Probing Algorithm for Metrology

Fatemeh Panahi[1], Aviv Adler[2], A. Frank van der Stappen[3] and Ken Goldberg[4]

***Abstract*— Metrology, the theoretical and practical study of measurement, has applications in automated manufacturing, inspection, robotics, surveying, and healthcare. An important problem within metrology is how to interactively use a measuring device, or *probe*, to determine some geometric property of an unknown object; this problem is known as *geometric probing*. In this paper, we study a type of *proximity probe* which, given a point, returns the distance to the boundary of the object in question. We consider the case where the object is a convex polygon P in the plane, and the goal of the algorithm is to minimize the upper bound on the number of measurements necessary to exactly determine P. We show an algorithm which has an upper bound of $3.5n + k + 2$ measurements necessary, where n is the number of vertices and $k \leq 3$ the number of acute angles of P. Furthermore, we show that our algorithm requires $O(1)$ computations per probe, and hence $O(n)$ time to determine P.**

I. Introduction

Metrology, the study of measurement, has applications in manufacturing, inspection, robotics, surveying, and healthcare ([3], [4]). An important aspect of metrology is the problem of how to most efficiently use a given measurement device, or *probe*, to obtain a specific piece of complex information. When the measurement device and object of interest are geometric, the problem of obtaining information about the object through repeated use of the device is known as *geometric probing*. A common version of this problem is to deduce the shape of an unknown object using as few probes as possible.

Efficient algorithms for probing convex polytopes have been the subject of several papers, starting with Cole and Yap [7], who studied the complexity (in terms of number of probes required) of *Determining the Shape of an Unknown Convex Polygon* by using probes which travel along a straight line chosen by the algorithm and stop when they collide with the polygon (later referred to as *finger probes* [1], [8], [9]). A number of probe types and algorithms were presented by Dobkin et al. [9]. These include the finger probes previously studied by Cole and Yap; *hyperplane probes*, which consist of a hyperplane (whose angle is chosen by the algorithm) which sweeps over the whole space and stops when it collides with the polygon; and *silhouette probes* (also called *projection probes* [13]), which provide the projection of the polygon onto a chosen subspace. Other probes which have been studied for convex polygons include *x-ray probes* ([1], [8], [12]), which measure the length of intersection between a chosen line and the unknown polygon, and *half-plane probes* [14], which measure the area of intersection between a chosen half-plane and the unknown polygon.

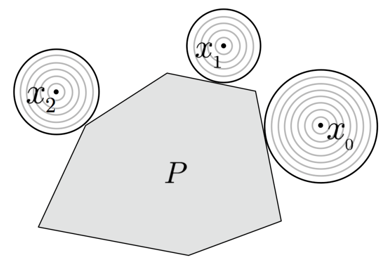

Fig. 1. An unknown polygon P with proximity probes at x_0, x_1, x_2

In this paper, we consider *proximity probes* which, given a point, return the distance to the boundary of the object in question. We consider the case where our object of interest is a convex polygon and our goal is to determine P exactly with the fewest probes possible. Fig. 1 illustrates an instance of our problem, with three proximity probes at x_0, x_1, x_2 respectively measuring an unknown convex polygon P.

This type of problem is relevant for situations where a relatively simple sensor must be used intelligently to efficiently extract a piece of complex information. For example, one application of our instance might be robotic exploration with non-directional sonar where echo time is proportional to distance. There are also possible relevancies to semiconductor manufacturing, where it can be valuable to inspect the precise shape of an etched silicon structure ([2], [15]). Notable techniques in current use include scanning probe microscopy (SPM) ([2]), which performs a continuous scan over the material with a physical probe, and virtual metrology (VM) ([5], [6]), which uses measurements of tool parameters during the production of wafers to statistically predict the final properties of the silicon. Our work suggests a new approach of interactively using a simple proximity probe a finite number of times to inspect these structures.

The remainder of the paper is organized as follows. In Section II, we introduce the problem and the definitions necessary for the algorithm. In Section III, we present our algorithm and analyze its complexity per probe; we also present a complete example of our algorithm for a simple polygon P. In Section IV we show an upper bound on the number of probes needed by our algorithm. Finally, in Section V, we summarize our results and discuss future work.

[1]Fatemeh Panahi is a Ph.D. candidate with the Department of Information and Computing Sciences, Utrecht University, Utrecht, Netherlands. She is supported by the Netherlands Organization for Scientific Research (NWO)

[2]Aviv Adler is an undergraduate student with the Department of Mathematics, Princeton University, Princeton NJ, USA

[3]A. Frank van der Stappen is with the Department of Information and Computing Sciences, Utrecht University, Utrecht, Netherlands

[4]Ken Goldberg is with the Departments of IEOR and EECS, UC Berkeley, Berkeley CA, USA

II. Problem Formulation and Preliminaries

We assume that all points and objects lie in the plane and that all positioning and measurements are exact.

For any two points or closed sets of points a, b, $\mathrm{dist}(a, b)$ denotes the Euclidean distance between a and b; for a lclosed subset S of the plane, $\partial(S)$ denotes its boundary, $\mathrm{Int}(S)$ denotes its interior, $\bar{S}$ denotes the closure of its complement (so both S and $\bar{S}$ contain $\partial(S)$), and $\mathrm{Conv}(S)$ denotes its convex hull. We also define *zero-disk* to mean a disk containing only its center.

In addition, for any disk of positive radius C and point z on its boundary, we define $L(C, z)$ to be the line tangent to C at z. We also define $H(C, z)$ to be the half-plane bordered by $L(C, z)$ which contains C, and $\bar{H}(C, z)$ to be the half-plane bordered by $L(C, z)$ which does not contain C.

A. Problem Formulation

Let P be an unknown convex polygon with n vertices and edges contained in a known disk D, and let the *probing function* f_P be defined over the the plane as

$$f_P(x) = \begin{cases} \mathrm{dist}(x, P) & : x \notin \mathrm{Int}(P) \\ -1 & : x \in \mathrm{Int}(P) \end{cases}$$

The probing algorithm is not explicitly given this function, but is allowed to call it as many times as necessary to find P exactly; the goal of this paper is to find an algorithm which minimizes the upper bound of probes necessary (and allows the next probe to be efficiently computed at each step). The points x for which it calls the function f_P are the *probes*, and the disks of radius $f_P(x)$ centered at these points are the *probe disks*, abbreviated as *p-disks* (by convention, if $f_P(x) = -1$ then no disk is produced). Every p-disk is by definition incident to P at exactly one point.

B. Condensed Probe Disks

Suppose we have two (distinct) p-disks C_a, C_b such that $C_a \subset C_b$. Since they both must be incident to P at exactly one point, they must be incident to P at the same point (otherwise it is impossible for one to contain the other); this point will by definition be the only point in $\partial(C_a) \cap \partial(C_b)$, which we call $p_{a,b}$. Furthermore, P must be interior disjoint with the half-plane $H(C_b, p_{a,b})$ since P is convex, $p_{a,b} \in C_b, P$, and C_b is not a zero-disk (because $\emptyset \neq C_a \subset C_b$). We thus define the *condense* operation on C_a, C_b which outputs $p_{a,b}$ as a zero-disk and associates with it the half-plane $H(C_b, p_{a,b})$; the products of this operation are called *condensed probe disks* (abbreviated as *cp-disks*). Furthermore, any p-disks which neither contain nor are contained by other p-disks (and so cannot be used by the condense operation) are also considered to be cp-disks. Note that cp-disks, like p-disks, must have exactly one intersection point with P (since cp-disks are either p-disks or zero-disks produced by the condense operation). If C^* is a cp-disk produced by the condense operation (and hence C^* is a point), we let $H(C^*)$ be its associated half-plane and $L(C^*)$ be the line bordering $H(C^*)$.

A small note: it is possible for a p-disk C_a to be contained in several other p-disks, none of which are contained in each other; however, this can only happen when C_a is a zero-disk and also at a vertex of P. In these cases, C_a will be condensed with every disk containing it to produce multiple condensed cp-disks.

C. Clockwise Ordering of cp-Disks

For any cp-disk C^*, let $p(C^*)$ be its intersection point with P. We note that by imposing a clockwise direction on the boundary of P, we can impose a clockwise order on the set of $p(C^*)$ for all cp-disks C^* (it is possible for two cp-disks to have the same contact point on P; but this can only happen on vertices of P). This then imposes a clockwise (cyclic) ordering on the set of cp-disks, where if multiple cp-disks happen to have the same contact point with P, they can be ordered by the lines tangent to them at the common contact point (a zero-disk C^*_{zero} produced by the condense operation is considered to have the line $L(C^*_{zero})$ as its tangent; a zero-radius cp-disk not produced by the condense operation cannot share a contact point with another p-disk or cp-disk since it would be contained by the other disk and hence not be a cp-disk by definition).

From now on we will attach indices to the cp-disks indicating their order. Specificially, we will let X be the ordered set of cp-disks, and implicitly label the disks in X as $C^*_1, C^*_2, ..., C^*_\alpha$. Since the ordering of cp-disks is cyclic, we assume that additions and subtractions on indices are performed modulo the number of disks.

Remark: It should be noted that in general, given an arbitrarily probed set of cp-disks, prior knowledge of P is necessary to deduce their exact ordering by the above criteria, and thus the ordering cannot be used by the algorithm. However, we will show that our algorithm chooses probes in such a way that this labeling can always be determined exactly without any prior knowledge of P.

D. Shadow Sets

Suppose we have two cp-disks C^*_i, C^*_j; for both we define a counterclockwise direction on their boundary. We define the lines $L_{i,j}$ and $L'_{i,j}$ to be the lines tangent to both C^*_i and C^*_j such that

- for both lines, C^*_i and C^*_j lie on the same side
- $L_{i,j}$ is given a direction coinciding with the counter-clockwise direction imposed on the two cp-disks, while $L'_{i,j}$ is given a direction opposing the counterclockwise direction
- Both lines, in their given directions, intersect C^*_i before C^*_j.

Note that $L_{i,j}$ is the same line as $L'_{j,i}$ but with the opposite direction imposed on it.

We now define the rays $l_{i,j}, l'_{i,j}$ to be the rays respectively lying on $L_{i,j}, L'_{i,j}$ with their sources at the respective points of tangency with C^*_j. For a ray l, we define $H_{right}(l)$ to be the quarter plane lying directly to the right of the ray, and $H_{left}(l)$ is analogously defined.

We then define the *shadow set* cast by C_j^* with respect to C_i^* as

$$S_i(j) = C_j^* \cup (H_{left}(l_{i,j}) \cap H_{right}(l'_{i,j}))$$

This set cannot contain any point of $\mathrm{Int}(P)$, since P cannot have any point in $\mathrm{Int}(C_j^*)$, must be incident to C_i^*, and is convex; similarly, P cannot contain any point of $\mathrm{Int}(S_i(j))$.

Notice that the boundary of C_j^* is partly on the boundary of $S_i(j)$ and partly in its interior; since P cannot contain any point of $\mathrm{Int}(S_i(j))$, its point of intersection with C_j^* must be on the part of $\partial(C_j^*)$ which is also on $\partial(S_i(j))$. We call this the *feasible arc* C_i^* imposes on C_j^* and denote it $\zeta_i(j)$.

E. The Neighbor-Infeasible Region

We first define the set $S_{i-1}(i) \cup S_{i+1}(i)$ to be the *neighbor-shadow set* of C_i^* (abbreviated as *ns-set*), denoted as $S(i)$ for convenience. Similarly, we define the set $\zeta_{i-1}(i) \cap \zeta_{i+1}(i)$ to be the *neighbor-feasible arc* of C_i^*, abbreviated as *nf-arc*; we denote it as $\zeta(i)$ for convenience.

For cp-disks produced by the condense operation, we instead use $S(i)$ to refer to the half-plane $H(C_i^*)$. Note that since no cp-disk can be contained in $\mathrm{Int}(H(C_i^*))$, $H(C_i^*)$ is a superset of $S_{i-1}(i) \cup S_{i+1}(i)$ for these cp-disks.

The *neighbor-infeasible region* R can now be defined as

$$R = \bigcup_{i=1}^{m} S(i) \cup \bar{D}$$

Intuitively, for each C_i^*, we simply take the ns-set of C_i^*, the half-planes associated with all cp-disks generated by the condense function, and the complement of D (the disk which we were initially given as containing P). Since R is composed of these pieces, P must be entirely contained in (the closure of) the complement of R.

We will show later that our algorithm behaves in such a way that the complement of R (the *neighbor-feasible region*) is a single connected piece; we therefore will assume it to be the case now. The boundary of R will then be naturally split into the following two basic types of pieces, which we call *sections*:

1) arcs of the boundary of D
2) connected subsets of the boundaries of the sets $S(i)$; we denote $\partial(S(i)) \cap \partial(R)$ as $\partial_R(S(i))$

Note that the second type of section has two possibilities:

a) if C_i^* was not produced through the condense operation, $\partial_R(S(i))$ is naturally split into at most three pieces, namely

- the nf-arc $\zeta(i)$
- a segment of the ray $l_{i-1}(i)$ (which we will denote $l(i)$ for convenience)
- a segment of the ray $l'_{i+1}(i)$ (which we will denote $l'(i)$ for convenience)

The other two pieces of the boundary of $S(i)$, namely $l'_{i-1}(i)$ and $l_{i+1}(i)$ cannot lie on $\partial(R) = \partial(\bar{R})$ because P in that case would impose the wrong ordering of the cp-disks.

b) if C_i^* was produced through the condense operation, $\partial_R(S(i))$ is just $L(C_i^*)$

Remark: Although the neighbor-infeasible set R is interior disjoint with P by definition, it is not necessarily the case that it is the full set of all infeasible points, i.e. the points which, given the p-disks, can't be contained in P.

F. Confirmation of Vertices and Edges, and the Query Set

We say a point v is *confirmed* if by considering X it can be shown that v is a vertex of P, and we say a line L is confirmed if by considering X it can be shown that l contains an edge of P; an edge e of P is also referred to as confirmed if the line extending it is confirmed. Any vertices or edges of P which are not confirmed are called *unconfirmed*. The list of confirmed vertices is denoted V_c and the list of confirmed edges is denoted E_c.

Now we consider $\partial(R)$, as described above as a collection of pieces of the boundaries of the $S(i)$ and D. Since $\partial(R)$ is continuous, there will be points which lie on more than one of the specified sections. Some of these points will lie on confirmed vertices or edges of P. The ones which do not will be called the *query set* Q, from which we will always probe (except for the very first probe). Furthermore, we define the *preferred query set* Q^* to be the subset of Q which does not contain any intersection points between two p-disks.

To confirm a vertex or line, we need to count how many p-disks are incident to it; an easy way to compute this from the set of cp-disks is to count the number of cp-disks tangent to L, double-counting those produced by the condense operation (since they correspond to two p-disks). Note that this means the number of cp-disks involved is at most the number of p-disks involved.

Furthermore, note that the set of all cp-disks passing through a point or tangent to a line must be consecutive.

We can confirm a point v as a vertex of P in these cases:

- if 3 p-disks pass through v
- if v is probed and $f_P(v) = 0$ (this implies that $v \in \partial(P)$; the fact that v was in Q, which is a necessary condition for being probed by the algorithm, means that v sits in a corner of R and thus cannot be in the middle of an edge of P, meaning it must be a vertex of P)
- if a segment of (confirmed or unconfirmed) line L on $\partial(R)$ and two p-disks touch v
- if segments of (confirmed or unconfirmed) lines L, L' on $\partial(R)$ and one p-disk touch v

If we confirm a vertex on a previously unconfirmed line, we can automatically confirm the line as well.

Additionally, we can confirm a line L as containing an edge of P if L is tangent to three p-disks. The cp-disks representing these three p-disks will necessarily be consecutive in X because they all have contact points with P on the same edge (and no other cp-disks will have contact points in the interior of this edge, since in that case L would have been confirmed earlier), and so given a cp-disk C_i^* we just need to check the three consecutive triples containing it.

In addition, if line L is tangent to two p-disks and passes through the intersection point v of the boundaries of two other p-disks, then both L and v can be confirmed. Also, if L is tangent to a p-disk and goes through the intersections of the boundaries of two different pairs of p-disks (call these points v_1, v_2), we can confirm v_1, v_2 and L.

Whenever a vertex v is confirmed, it automatically implies that probing v would return $f_P(v) = 0$; this means we can place a p-disk there *without* explicitly probing it, and perform the condense operation with any existing p-disks which happen to contain v. Since they all have the same contact point v with P, they will be consecutive in X, and later on we will show that there cannot be more than 3 such disks for any v, so this process takes constant time.

Similarly, whenever a line L is confirmed, we always have at least one, and often more than one, cp-disks tangent to L; at each tangent point x we know that $f_P(x) = 0$ so we may place a p-disk there without actually executing the probe function, and perform the condense operation with the original tangent p-disk to create a new cp-disk. Since an edge is always confirmed if it is incident to 3 cp-disks, the number of condense operations we need to perform is at most 3 for each confirmed line; thus this process takes constant time.

Remark: Thanks to the fact that we use the condense operation when we confirm vertices and edges (without requiring new probes), the lines corresponding to these condense operations are automatically incorporated into $\partial(R)$.

III. The Algorithm

We now present an efficient algorithm for solving the probing problem described in Section II. The algorithm maintains the circular ordered list X of cp-disks, sorted in clockwise order of their intersection point with P around $\partial(P)$, an algebraic representation of the neighbor-infeasible region R, lists of the confirmed vertices (V_c) and edges (E_c) of P, and representations of the query set Q and preferred query set Q^*. We present it in two parts: the first dealing with how to generate the next probe given X, R, V_c, E_c, Q, and Q^*, and the second dealing with how to update these objects given a new probe result. The algorithm terminates once (a) at least one vertex and edge have been confirmed and (b) every confirmed vertex is on two confirmed lines and every confirmed line contains two confirmed vertices.

In addition, a some extra information and pointers will be stored in these lists in order to allow the algorithm to execute all the steps in constant time, most notably pointers in Q for each element which point to its neighbors (in both X and Q); however, we omit the exact details.

A. Algorithm for Generating New Probes

The algorithm for generating new probes is divided into two distinct phases (preceded by a one-probe initialization): in Phase 1, we probe arbitrarily from the preferred query set Q^* when possible; when it is not, we choose instead from Q (both Q^* and Q are by definition a subset of the boundary of R) until some edge is confirmed; in Phase 2 (once an edge is confirmed), we probe points designed to confirm the vertices and edges of P in (roughly) clockwise order.

We also add the following definitions for reference in the algorithm:

- the first edge of P to be confirmed is denoted e_1 (i.e. the edge contained by the first line confirmed)
- the edges and vertices of P in clockwise order are $e_1, v_1, e_2, v_2, ..., e_n, v_n$
- for any edge e_i, we let L_i^* be the line containing e_i; note that it is the lines, not the edges themselves, which are directly confirmed by the algorithm
- at any given step of the algorithm, we let t be the largest index such that $e_1, v_1, e_2, v_2, ..., e_{t-1}$ are all confirmed (we can determine t from E_c and V_c without any extra direct knowledge of P)
- l is a ray originating on some point on e_{t-1} which we know is in P (for all $t > 2$, we use v_{t-2}; otherwise we use the contact point of some p-disk with the confirmed line containing e_{t-1}) and extending e_{t-1} in the direction coinciding with the clockwise direction around the boundary of P (this direction is also determinable from E_c and X without any extra knowledge of P)
- for any set S and ray γ, let $\rho(\gamma, S)$ be the furthest point along γ which is also in S

At the start, X, V_c, E_c, Q, Q^* are empty and $R = \bar{D}$, so we simply probe from an arbitrary point on the boundary of D. Because $P \subset \mathrm{Int}(D)$, this disk will have positive radius; because it is the first p-disk, it cannot be condensed and is thus also a cp-disk. In addition, it will not have any neighbors in X since it is the only disk in X, so its shadow set is by convention defined to be itself. Thus, R is simply the union of this disk and the complement of D, and the boundary of R will consist of an arc of this disk plus an arc of D. Hence, by definition, Q consists of the two points of intersection between the boundaries of D and the first cp-disk.

Algorithm Steps:

1) While no line has been confirmed, at each step we check if Q^* has at least one element. If it does, we choose an arbitrary point $x \in Q^*$ and probe it; if not, we choose an arbitrary point $x \in Q$ and probe it.
2) Once a line has been confirmed, we let the edges and vertices of P, the index t, and the ray l be defined as above. We repeat the following step until both e_t and v_{t-1} are confirmed (at which point, by definition, the index t increases, and we start Phase 2 again; we terminate once v_t is confirmed on e_1).
   Let $x = \rho(l, \bar{R})$; an intuitive idea of x is that it is the furthest clockwise point on the confirmed line containing e_{t-1} which is not in the neighbor-infeasible region R. We note then that since x is the furthest point on $l \subset L_{t-1}^*$, it must also be on some other object on the boundary of R; hence, either $x \in V_c$ (if x happens to be v_{t-1} and is already confirmed) or $x \in Q$.
   If $x \in Q$ then it must be both on L_{t-1}^* and some other piece of the boundary of R. In particular, it can be on the following
   - an nf-arc $\zeta(i)$ of some
   - another confirmed line
   - an unconfirmed line, either corresponding to the

output of a condense function or incident to two (consecutive) cp-disks
- the boundary of D

We then do the following:

a) if $x \in V_c$, call *Next Edge*
b) if $x \in Q$ and $x \notin \zeta(i)$ for all i, probe x
c) if $x \in Q$ and $x \in \zeta(i)$ for some i, then it is one endpoint of the arc $\zeta(i) \cap \partial(R)$; let x' be the other endpoint. This point by definition will either be x's neighbor in Q or will be an endpoint of $\zeta(i)$, and hence is retrievable in constant time

Remark: Although in Phase 1 we are allowed to probe any $x \in Q^*$ (or, if Q^* is empty, any $z \in Q$) at each step, if we wish to minimize the time complexity of choosing the next probe at each step, we need a retrieval method which produces a member of Q^* or Q in constant time; having either a stack or a queue as an additional data structure for Q^* and Q are the most natural ways of achieving this.

The Next Edge Procedure

This procedure is called when e_{t-1} and v_{t-1} are both confirmed but e_t is not confirmed. Let us consider the set of cp-disks incident to v_{t-1}; they will be consecutive in X, and will have been produced by the condense function (at the moment that v_{t-1} was confirmed). Let C_i^* be the last cp-disk among them; let $\mathrm{N}_Q(C_i^*)$ be C_i^*'s next neighbor (in the clockwise direction) in Q. We then probe $\mathrm{N}_Q(C_i^*)$ (updating the maintained information as we go so i and $\mathrm{N}_Q(C_i^*)$ can change after each probe) until the next edge is confirmed, at which point t can be updated and we return to the main loop of Phase 2. We note that $\mathrm{N}_Q(C_i^*)$ is actually the point on $L(C_i^*)$ furthest from v_{t-1}.

The Pseudocode

For the pseudocode, we introduce some extra notation and functions (and show, where necessary, that these functions can be computed efficiently). We define the sets E_c^*, V_c^* to be respectively the subset of E_c consisting of those lines which do not contain two points from V_c, and the subset of V_c consisting of those points which are not contained by two lines from E_c. Intuitively, E_c^* and V_c^* consist of the confirmed lines and vertices whose adjacent vertices and lines, respectively, have not been confirmed yet. These sets are easy to maintain with flags attached to both E_c and V_c.

For the case (c) of Phase 2, if $x \in \zeta(i)$, then we denote the other endpoint of the arc $\zeta(i) \cap \partial(R)$ as $q(x)$.

For any $x \in Q$, we note that since we can retrieve its neighbors in X in constant time, we can determine whether it is on some nf-arc in constant time; we will treat this as a binary valued funtion $\mathrm{nf}(x)$ which is $true$ when x is on some nf-arc, and $false$ otherwise.

The *RandomElement* function refers to random or arbitrary choice of some element from a set; the *Probe* function refers to the full update algorithm (described in Section IIIB), which uses and modifies all the objects in the program. Most object updates occur within the Probe function.

Note that by the time Phase 2 starts, by definition, we will have at least one member of E_c; note also that maintaining Q^* is only necessary for Phase 1.

Algorithm 1 Identifying P using proximity probes

```
procedure DETERMINEP(D)
    V_c, E_c ← null                                  ▷ Initialization
    ∂(R) ← ∂(D)
    x ← RandomElement(∂(D))
    run Probe(x)
    while E_c = null do                              ▷ Phase 1
        if Q* ≠ null then
            x ← RandomElement(Q*)
        else
            x ← RandomElement(Q)
        run Probe(x)
    while E_c* ≠ null and V_c* ≠ null                ▷ Phase 2
        x ← ρ(L, R)
        if x ∈ V_c then                              ▷ Case a:
            run NextEdge(x)                          ▷ x = v_{t-1}
        else if nf(x) = false then                   ▷ Case b:
            run Probe(x)                             ▷ x is not on an nf-arc
        else                                         ▷ Case c:
            x' ← q(x)                                ▷ x is on an nf-arc
            run Probe(x')
    return V_c                                       ▷ Return P as a set of vertices
end procedure

procedure NEXTEDGE(x)
    while ¬∃e ∈ (E_c\e_{t-1})|x ∈ e
        x' ← N_Q(C_i*)
        run Probe(x')
end procedure
```

B. Algorithm for Handling a New Probe

The algorithm for updating the maintained information (X, R, E_c, V_c, Q, Q^*) is relatively simple since we usually probe from the set Q (since $Q^* \subset Q$). To update X in this case, we merely note that each point $x \in Q$ is specifically linked to two consecutive 'neighbors' in X.

If the new p-disk contains or is contained by one or both of the 'neighbor' cp-disks of its center, we perform the condense operation; this check trivially takes constant time since it has only two neighbors. It cannot contain or be contained by any non-neighboring cp-disks, and therefore checking whether the condense operation has to be used has constant time complexity per step.

The only case where we do not probe from Q is in Phase 2, when line L_{t-1}^* containing edge e_{t-1} is meets $\zeta(i)$ (by definition at an endpoint of $\zeta(i) \cap \partial(R)$) and, in addition, the other endpoint of $\zeta(i) \cap \partial(R)$ is not in Q. Even if we cannot determine it from our observations alone, our original definition of the ordering (depending on P) is still valid; because the new disk has its center on the neighbor-feasible arc of C_i^*, it must be a neighbor of C_i^*. Furthermore, since it is the other (further clockwise around the boundary of $\bar{R}$)

endpoint of $\zeta(i) \cap \partial(R)$, the remaining set of points at which C_i^* can be incident to P, which is a subset of $\zeta(i) \cap \partial(R)$, is counterclockwise from all points of the new disk (around the boundary of $\bar{R}$). Hence, the new disk cannot be between C_{i-1}^*, C_i^* and can be inserted between C_i^*, C_{i+1}^*.

The remainder of the updates involve updating V_c and E_c, and in turn updating Q to not include confirmed vertices or edges; as any vertex or line is automatically confirmed when three p-disks are tangent to it, and thus these checks remain in constant time. Updating the relevant stored information is constant for each element of R, Q, Q^*, V_c, E_c and X we update, and for each set only a bounded number of elements (the neighbors of the probed point) are updated, so the total updating time has complexity $O(1)$ per probe.

C. Example

Here we present a simple example (Fig. 2) of our algorithm determining a polygon P with four vertices, one of which is acute (so $n = 4, k = 1$). Probes are represented by filled dots and labeled in order (starting from x_0); cp-disks are shown by black circles (with cp-disks which were condensed shown by dashed white lines). In Phase 1 of the algorithm Q^* is denoted by empty dots; in Phase 2, the next probe is denoted by an empty dot.

IV. Bounding the Required Number of Probes

We first establish the following notation. Let v be a vertex of P; we then write $\angle_P(v)$ to refer to the angle of P at v. If v is confirmed, we note that this means the algorithm would have condensed the disks incident to v, so that $\bar{R}$ would have an angle at v; we write $\angle_R(v)$ to refer to this angle.

Note that $\angle_P(v)$ is always contained in $\angle_R(v)$ and that $\angle_R(v)$ never increases as the algorithm goes on.

We omit the proofs of the following lemmas; however, the interested reader can find them in our technical report [16].

A. Preliminary Lemmas

Lemma 4.1: Assume that v is the intersection point on $\partial(R)$ of the boundaries of two p-disks C_i and C_j, neither of which contains the other. If we probe from $x \in \bar{R}$ such that $x \neq v$, the resulting p-disk C cannot pass through v unless $\angle_P(v)$ is acute. Additionally, if $\angle_P(v)$ is acute and C passes through v, then $\angle_R(v)$ becomes acute.

Lemma 4.2: Let v be a confirmed vertex of P, and let $x \in \bar{R}$ be the next probed point which produces a disk C (v is already confirmed, so $x \neq v$ since we don't probe confirmed vertices). Then C can be incident to v only if $\angle_P(v)$ is acute, $\angle_R(v)$ is not acute; futhermore, afterwards, $\angle_R(v)$ will be acute (so no new p-disk can be incident to v).

Corollary 4.3: Let v be a vertex of P such that when v is confirmed, it is *not* by being probed directly. Then, when the algorithm finishes,

- if $\angle_P(v)$ is not acute, the number of p-disks incident to it is at most 2
- if $\angle_P(v)$ is acute, the number of p-disks incident to it as at most 3

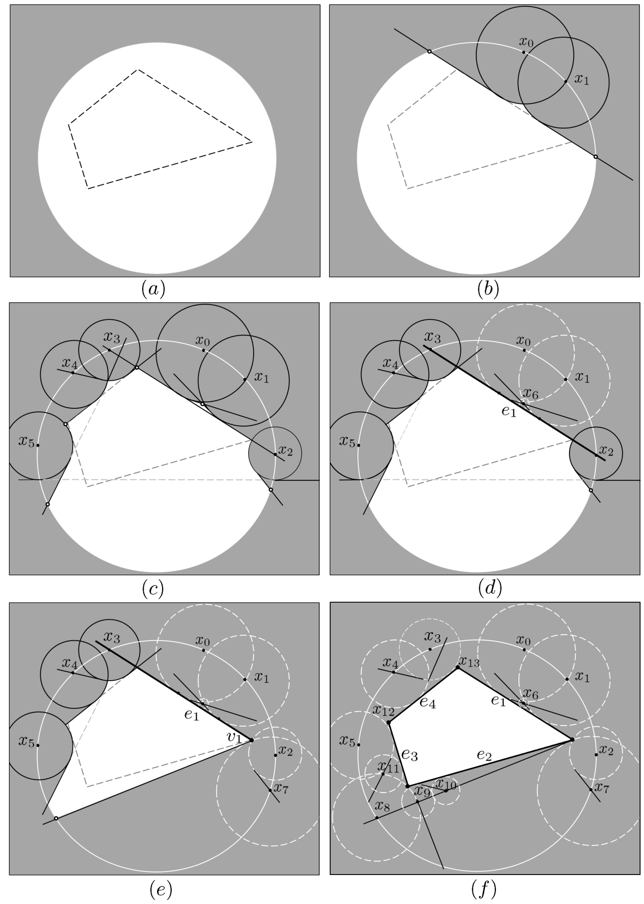

Fig. 2. *(a)* An example of a quadrilateral with one acute-angle vertex which is contained in a known disk. Let the acute angle be denoted by v_1, and the other vertices are labeled in clockwise order, as per the notation used in the algorithm; *(b)* x_0 is an arbitrary point on the boundary of R and x_1 is one of the intersection points of the disk resulting from x_0 and $\partial(R)$; *(c)* Illustration of all probes but one of Phase 1 of the algorithm; *(d)* After seven probes the first edge is confirmed, and the disks incident to that edge are then condensed; *(e)* In Phase 2 of the algorithm, case (c) of the algorithm occurs, resulting in a probe at x_7. This confirms v_1, and therefore we apply the condense operation to the cp-disks centered at x_2 and x_7. It can be observed that two p-disks are incident to v_1 (which is the acute angle) i.e. $\omega(v_1) = 2$; *(f)* After 14 probes, P has been determined

Lemma 4.4: Let e be a confirmed edge and v be one of its endpoints. Let $x \in \bar{R}$ such that x doesn't lie on the line extending e. If we probe from x, the resulting disk cannot be incident to v unless v is an acute angle vertex of P.

Remark: We note that as long as we only probe from points in $\partial(R)$ which are not confirmed vertices or in the interior of any line segment on $\partial(R)$ contained by a confirmed line, we will never create a p-disk which will be incident to the interior of any previously confirmed edge.

B. Undesirable Confirmations

The bounds derived in the previous section are only violated (by 1) if v is confirmed while incident to three p-disks, one of which is the zero-disk centered at v itself (this applies regardless of whether $\angle_P(v)$ is acute). However, we note that if one of the two non-zero p-disks is also tangent to one of the edges of P adjacent to v, we may associate it with that edge instead (so that the bound is not considered violated), and hence need only worry about the possibility that neither of the non-zero p-disks are tangent to an adjacent edge. We call such cases *undesirable confirmations*.

Lemma 4.5: Let m be the number of undesirable confirmations which occur over the course of the algorithm. Then $m \leq n/2 + 1$.

C. Analysis of the Algorithm

We now wish to find an upper bound for the number of probes used by our algorithm; this is achieved by analyzing the number of p-disks that can be incident to any edge or vertex of P when it is confirmed. We now assume that no undesirable confirmation occurs; later, we will note that by Lemma 4.5, each undesirable confirmation adds at most one probe to the upper bound, and that the number m of undesirable confirmations is bounded above by $n/2+1$, and add this to the bound we derived.

At any given step in the algorithm, let $\phi(e)$ and $\phi(v)$ denote the number of p-disks incident to unconfirmed edge e and unconfirmed vertex v respectively; and let $\omega(e)$ and $\omega(v)$ denote the number of p-disks which are incident to confirmed edge e and confirmed vertex v, respectively.

We first consider the number of p-disks any object can have adjacent to it at the moment it is first confirmed; by convention, if a p-disk is incident to both some confirmed vertex and some confirmed line(s) (if it is a zero-disk, it can be incident to a vertex and two lines), we associate it with the vertex only. We perform this analysis on the two basic phases of the algorithm.

For Phase 1 (i.e. confirming the first edge), there are two possible cases for the number of probes which will suffice to confirm the first edge e_1 with clockwise endpoint v_1:

- If $\phi(v_1) \leq 1$ three disks are sufficient to confirm e_1.
- If v_1 is confirmed or $\phi(v_1) = 2$, then two disks are sufficient to confirm e_1.

We will conduct the same analysis for Phase 2 by computing the possible values of $\omega(v_{i-1})$ and $\omega(e_i)$ when they are first confirmed (which depends on whether v_{i-1} is acute or not) for $1 < i \leq n$. We note that no vertex can be confirmed on $\partial(D)$ because $P \in \mathrm{Int}(D)$.

Case 1: v_{i-1} is not confirmed and $\phi(v_i) \leq 1$. Since v_{i-1} is not confirmed but e_{i-1} is confirmed, $\phi(v_{i-1}) \leq 1$. We consider the two possible sub-cases: either v_{i-1} is not an acute angle vertex of P, or it is.

- Suppose v_{i-1} is not an acute angle vertex. It could either have been confirmed by case (b) or case (c) from Phase 2 of the algorithm.
  - Suppose it was confirmed by case (b); let x be the point probed. For case (b) of the algorithm to confirm a vertex, the result of the probe must be 0 (i.e. $f_P(x) = 0$), and this new zero-disk is the only disk incident to v_{i-1}; thus $\omega(v_{i-1}) = 1$. In this case, x (which is actually v_{i-1}) cannot lie on the boundary of D (as in this case $x \in P \subset \mathrm{Int}(D)$), so x is on a segment of an (confirmed or unconfirmed) line L on $\partial(R)$; this line will then be confirmed as e_i with $\omega(e_i) = 2$.
  - Suppose it was confirmed by case (c). By Lemma 4.4, the new p-disk cannot pass through v_{i-1}, so $\omega(v_{i-1}) = 1$. We observe that to confirm v_{i-1}, the new p-disk must reduce the feasible arc of the previous p-disk containing v_{i-1} to a single point; to do this, it must confirm e_i. Hence, since $\omega(v_{i-1}) = 1$ and $\phi(v_i) \leq 1$, we get $\omega(e_i) = 2$.

  Therefore, in all cases, $\omega(v_{i-1}) = 1$ and $\omega(e_i) = 2$.
- If v_{i-1} is an acute angle vertex. This is similar to the above case, except that as Lemma 4.4 doesn't hold for acute angles, we include the possibility that in case (c) the resulting p-disk will pass through v_{i-1}. If so, v_{i-1} is confirmed, and the next iteration of the algorithm will be case (a). As $\phi(v_i) \leq 1$, $\omega(e_i) = 2$, and when v_{i-1} is confirmed in the next iteration $\omega(v_{i-1}) \leq 2$.

Case 2: v_{i-1} is not confirmed and either $\phi(v_i) = 2$ or v_i is confirmed. This case is similar to case 1, except that because $\phi(v_i) = 2$ (or v_i is confirmed), e_i is incident to at most one disk, and v_{i-1} will be confirmed immediately after e_i is confirmed. So, $\omega(e_i) = 1$ and $\omega(v_{i-1}) \leq 2$, if v_{i-1} is acute and $\omega(v_{i-1}) = 1$ if it is not.

Case 3: v_{i-1} is confirmed and $0 \leq \phi(v_i) \leq 1$. We consider the two possible sub-cases: either v_{i-1} is not an acute angle vertex of P, or it is.

- v_{i-1} is not an acute angle vertex. Since v_{i-1} is confirmed before e_i and v_{i-1} is not an acute angle, by Lemma 4.4, $\omega(v_{i-1}) = 2$, and case (a) will immediately follow in the algorithm. The next edge e_i will be confirmed by two incident disks since $\phi(v_i) \leq 1$, so $\omega(e_i) = 2$.
- v_{i-1} is an acute angle vertex. According to Lemma 4.4, it is possible that v_{i-1} has been confirmed with three disks as v_{i-1} is an acute angle. Therefore, $\omega(v_{i-1}) \leq 3$. As in the previous case, $\omega(e_i) = 2$.

Case 4: v_{i-1} is confirmed and either $\phi(v_i) = 2$ or v_i is confirmed. We again consider the same two possible sub-cases as in the above cases.

- v_{i-1} is not an acute angle vertex. As in case 3, $\omega(v_{i-1}) = 2$, but the next edge will be confirmed with one incident disks since v_i is incident to more than one disk (or already confirmed), so $\omega(e_i) = 1$
- v_{i-1} is an acute angle vertex. As in case 3, $\omega(v_{i-1}) \leq 3$, and $\omega(e_i) = 1$ since v_i is incident to multiple disks (or already confirmed).

Finally, it is clear that v_n will be confirmed with one disk. Table I summarizes the result for the above four cases.

Theorem 4.6: Our algorithm uses at most $3n+m+k+1 \leq 3.5n + k + 2$ probes to find P, where $k \leq 3$ is the number of acute angles of P; each probe is computed in $O(1)$ time, thus leading to an overall time complexity of $O(n)$.

Proof: We note that no p-disk generated at any point by the algorithm can be incident to a previously-confirmed edge or to a previously-confirmed non-acute angle vertex once both edges adjacent to it have been confirmed. Note also that since the algorithm never probes from the interior of $\bar{R}$, the algorithm never uses a probe which returns -1. Therefore, the number of probes needed is equal to the sum of the number of p-disks incident to each edge and vertex of P when they are confirmed, with the possible additional k for the acute angles already taken care of by assuming the worst case at time of confirmation. Let n_j be the number of times case j occurs, and k_j be the number of times case j occurs with an acute vertex; then $\sum_{j=1}^{4} n_j = n-1$ and $\sum_{j=1}^{4} k_j \leq k$ since the cases begin once e_1 is confirmed.

We now consider the number of p-disks incident to each edge and vertex of P when they are confirmed, assuming no undesirable confirmations:

- e_1 is incident to at most 3 p-disks when it is confirmed
- For $j = 1, 4$, by Table I we note that $\omega(v_{i-1})+\omega(e_i) \leq 4$ if v_{i-1} is acute, and $\omega(v_{i-1}) + \omega(e_i) = 3$; hence at most $3n_j + k_j$ probes were used.
- For $j = 2$, by Table I we note that $\omega(v_{i-1})+\omega(e_i) \leq 3$ if v_{i-1} is acute, and $\omega(v_{i-1}) + \omega(e_i) = 2$; hence at most $2n_2 + k_2$ probes were used
- For $j = 3$, by Table I we note that $\omega(v_{i-1})+\omega(e_i) \leq 5$ if v_{i-1} is acute, and $\omega(v_{i-1}) + \omega(e_i) = 4$; hence at most $4n_3 + k_3$ probes were used

Consider what happens in case 3 (with vertex v_{i-1} and edge e_i); it occurs when v_{i-1} is incident to two disks (or is confirmed) before e_i is confirmed. If $i = 2$, then e_1 must have been adjacent to 2 p-disks. If $i > 2$, then case 3 was preceded by either case 2 or case 4; if it was case 4, then since v_{i-1} was already confirmed, e_{i-1} must have been confirmed with one fewer p-disk than our above bounds.

Thus, every instance of case 3 (which requires one more probe per vertex-edge pair than cases 1 or 4), there is a corresponding instance either of case 2 (which requires one fewer probe per vertex-edge pair than cases 1 or 4) or of case 4 (or the base case) in which at least one fewer probe was used than the bound above. So, since case 3 is the only case in which more probes are required than cases 1 and 4, and since we showed that every instance of case 3 is 'offset', we can bound the total number of probes needed by the number needed if only cases 1 and 4 occurred.

Thus, the pairs $(v_1, e_2), ..., (v_{n-1}, e_n)$ plus e_1 require at most $3n + k$ probes to confirm; the final vertex v_n requires one more, giving an upper bound of $3n + k + 1$ probes with the assumption that no undesirable confirmations occurred. Each undesirable case increases the upper bound by at most 1, and the number of such cases (by Lemma 4.5) is $m \leq n/2 + 1$. Hence, we compute our true upper bound as $3n + m + k + 1 \leq 3.5n + k + 2$ probes. Finally, we note that in Section III(B) we showed that each probe requires $O(1)$ time computation, and therefore the total computation time required by the algorithm is $O(n)$. ■

V. Conclusion and Future Work

In this paper, we defined a type of proximity probe and showed an algorithim which finds the shape an unknown convex polygon P (with n vertices, $k \leq 3$ of which are acute angle vertices) requiring at most $3.5n + k + 2$ probes, with each probe requiring $O(1)$ time to compute.

In future work we will explore extending these results to 3 and higher dimensions, and to the case where measurements are not precise and lie within some bounds of the true value, which may permit bounding the shape of an unknown object. We will also look at the alternative problem, introduced by Goldberg and Rao [11], of identifying the object P from a finite set of possible objects by probing. Finally, we will consider non-convex objects inspired by the approach that Boissonnat and Yvinec [10] developed to extend finger probes to non-convex polyhedra, and study the problem of using distance probes from inside the polygon.

TABLE I
$\omega(v_{i-1}), \omega(e_i)$ FOR $1 < i \leq n$

Case	v_{i-1}	v_i	v_{i-1}: Not acute $\omega(v_{i-1}), \omega(e_i)$	v_{i-1}: acute $\omega(v_{i-1}), \omega(e_i)$
1	NC	NC, $\phi(v_i) \leq 1$	$1, 2$	$\leq 2, 2$
2	NC	C or $\phi(v_i) = 2$	$1, 1$	$\leq 2, 1$
3	C	NC, $\phi(v_i) \leq 1$	$2, 2$	$\leq 3, 2$
4	C	C or $\phi(v_i) = 2$	$2, 1$	$\leq 3, 1$

References

[1] Skiena, S. S., Problems in geometric probing. Algorithmica, 4(4):599-605,1989.

[2] Sergei V. K.; Alexei, G, Scanning Probe Microscopy of Functional Materials, 2011.

[3] Dotson, Connie L., Fundamentals of Dimensional Metrology, 2006.

[4] Czichos, H., Saito, T., and Smith, L. E., Springer Handbook of Metrology and Testing, 2011.

[5] Susto, Gian A., et al., An Information-Theory and Virtual Metrology-based approach to Run-to-Run Semiconductor Manufacturing Control. Proceedings of the 8th IEEE International Conference on Automation Science and Engineering, 358-363, August 2012

[6] Pampuri, Simone, et al., Multistep Virtual Metrology Approaches for Semiconductor Manufacturing Processes. Proceedings of the 8th IEEE International Conference on Automation Science and Engineering, 91-96, August 2012

[7] Cole, R. and Yap, C. K.. Shape from probing. Journal of Algorithms, 8(1):19-38, 1987.

[8] Skiena S. S., Interactive Reconstruction via Geometric Probing, Proceedings of the IEEE 80, 1364-1383, 1992.

[9] Dobkin, D., Edelsbrunner, H., and Yap, C. K., Probing convex polytopes. In Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, 424-432, Berkeley, California, 1986.

[10] Boissonnat, J. D. and Yvinec, M., Probing a scene of nonconvex polyhedra. Algorithmica, 8:321-342, 1992.

[11] Rao, A. S. and Goldberg, K. Y., Shape from diameter: Recognizing polygonal parts with a parallel-jaw gripper. Intl. J. of Robotics Research, 13(1):16-37, 1994.

[12] Meijer, Henk and Skiena, Steven S., Reconstructing Polygons from X-Rays, Geometriae Dedicata, 61(2), pp 191-20, 1996.

[13] Li, S.-Y. R., Reconstruction of polygons from projections. Information Processing Letters, 28:235-240, 1988.

[14] Skiena S. S., Probing Convex Polygons with Half-Planes, Journal of Algorithms 12, 359-374, 1991.

[15] Niemann, James, Electrical Measurements on Nanoscale Materials. Keithley Instruments tutorial paper, 2004.

[16] F. Panahi, A. Adler, A.F. van der Stappen, K. Goldberg, Efficient Distance Probing Algorithms for Metrology, Technical Report UU-CS-2013-010, Dept. of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands (2013).