Computing the Constrained Euclidean, Geodesic and Link Centre of a Simple Polygon with Applications

Prosenjit Bose, Godfried Toussaint

Abstract— **In the manufacturing industry, finding a suitable location for the *pin gate* (the pin gate is the point from which liquid is poured or injected into a mold) is a difficult problem when viewed from the fluid dynamics of the molding process. However, experience has shown that a suitable pin gate location possesses several geometric characteristics, namely the distance from the pin gate to any point in the mold should be small and the number of turns on the path from a point in the mold to the pin gate should be small. We address the problem of computing locations that possess these geometric characteristics. Given a mold M (modeled by an n vertex simple polygon) we show how to compute the Euclidean center of M constrained to lie in the interior of M or on the boundary of M in $O(n \log n + k)$ time where k is the number of intersections between M and the furthest point Voronoi diagram of the vertices of M. We show how to compute the geodesic center of M constrained to the boundary in $O(n \log n)$ time and the geodesic center of M constrained to lie in a polygonal region in $O(n(n+k))$ time. Finally, we show how to compute the link center of M constrained to the boundary of M in $O(n \log n)$ time.**

I. Introduction

In the manufacturing industry, two of the most popular classes of production methods are injection molding and gravity casting. Each of the two methods produces an object by filling a mold or cast of the given object with a liquid, and removing the object once the liquid has hardened. (see Figure 1). The difference between the two methods is that liquid is injected into the mold in injection molding whereas liquid is poured into the mold and gravity is the sole force acting on the liquid in gravity casting [11]. A mold or cast, as defined in [5], refers to the whole assembly of parts that make up a cavity into which liquid is poured to give the shape of the desired component when the liquid hardens.

In this paper, we consider the problem of determining a suitable location for the *pin gate*. The *pin gate* is the point on the mold from which the liquid is poured or injected into the cavity. The location of the pin gate plays an important role in determining whether or not an object built by one of the two manufacturing processes will have surface defects.

P. Bose's research was supported by a Killam and an NSERC postdoctorate fellowship. Département de Mathématiques et d'Informatique, Université du Québec à Trois-Rivières, C.P. 500, Trois-Rivieres, Québec G9A 5H7, Canada. This work was completed while the author was at the School of Computer Science, McGill University and the Department of Computer Science, University of British Columbia. E-mail: jit@uqtr.uquebec.ca.

G. Toussaint's research was supported by grants NSERC-OGP0009293 and FCAR-93ER0291. Address: School of Computer Science, McGill University, 3480 University, Montréal, Québec, H3A 2A7. E-mail: godfried @cs.mcgill.ca.

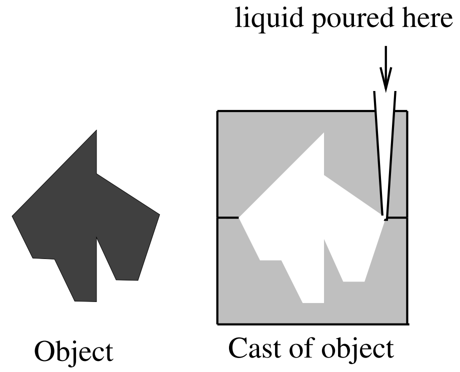

Fig. 1. An object and its cast.

Many factors play a role in determining a suitable location for the pin gate when considered from the point of view of fluid dynamics and physics of the whole molding process. To date, trial and error, guided by engineering experience, has been the main method in determining a suitable location for the pin gate [16], [22], [26]. However, through this experience, a few of the key characteristics of an ideal location for a pin gate have been uncovered.

> If the distance from the gate to the extremities of the mold cavity is too great, the metal freezes prematurely, and misruns result. [16]

This quote points out one of the key problems faced by cast designers. In order to avoid this problem, designers must place the pin gate at a location where the distance from it to the extremities of the mold cavity is not too great. Another key characteristic of casts that leads to surface defects is the presence of many "sharp corners or overhanging or protruding sections..." [1]. These "sharp corners" disrupt the flow of molten liquid leading to surface defects. Therefore, the pin gate must be placed in a location such that the flow of molten liquid from the gate does not encounter too many sharp corners or make too many turns. For an overview of the many other factors causing defects in molds and casts, the reader is referred to [16], [1].

These observations allow one to deduce the following properties for a good pin gate location:

Property 1: The maximum distance from the pin gate to any point in the object should be small.

Property 2: The maximum number of turns the liquid takes on its path from the pin gate to any point in the object should be small.

When viewed from a purely geometric perspective, these problems can indeed be solved optimally. The geometric solutions provide an initial approximation that can aid in the search for a suitable location. In this paper, we solve the pin gate location problem for molds modelled as simple polygons which find applications in polymer molds. In practice, many 3-dimensional objects are almost flat so that in effect they can be considered as 2-dimensional. Therefore the 2-dimensional theory is more important than may appear at first glance, and sheds some light on the 3-dimensional problem.

The two properties that a pin gate should satisfy have several geometric interpretations. Property 1 can be interpreted as the point inside the simple polygon whose maximum distance to any point in the object is minimized. If distance is measured in the Euclidean metric, this point is referred to as the constrained Euclidean center. Sometimes a pin gate is constrained to lie on the boundary of the mold. In such a case, Property 1 can be interpreted as the point on the boundary of the simple polygon whose maximum distance to any point in the polygon is minimized with respect to all points on the boundary. This point is referred to as the *boundary-constrained* Euclidean center. On the other hand, distance can be measured by the geodesic metric, i.e., the minimum distance the liquid must travel inside the mold to reach a destination. In this case, Property 1 places the pin gate at the geodesic center, which by definition is constrained to lie inside the polygon, and the *boundary-constrained* geodesic center, respectively.

Property 2 can be interpreted as the link metric. The link metric measures the number of turns in a path between two points. For example, if two points can be joined by a line segment, then they are at link distance 1. The points inside a simple polygon, whose link distance to any other point in the polygon is minimized, are referred to as the link center. If the pin gate is constrained to the boundary, then it is referred to as the *boundary-constrained* link center.

As this is only an extended abstract, many proofs and details are omitted. The reader is referred to [7] for the details.

II. Constrained Euclidean Center

In this section, we show how to find the point inside a simple polygon P as well as the point on ∂P whose maximum Euclidean distance to every point of P is minimized. These points are known as the *Euclidean center constrained to lie in the polygon*, and the *Euclidean center constrained to lie on the boundary of the polygon*, respectively.

The Euclidean center of the vertices of a simple polygon may be a good candidate for the location of the pin gate, but the center might lie outside the polygon. Therefore, the location of the center must be constrained to lie inside the polygon or on its boundary since otherwise it cannot serve as a pin gate. Therefore, given an object modelled as a simple n vertex polygon, we wish to find the point lying inside the polygon whose maximum Euclidean distance to any point is minimized with respect to all points in the polygon. Since the furthest neighbor of a point must be a vertex, we can restrict our attention to finding the point lying inside the polygon whose maximum Euclidean distance to any vertex is minimized with respect to all points in the polygon. We also want the point on the boundary whose maximum Euclidean distance to any vertex is minimized with respect to all points on the boundary. Although the Euclidean center is unique, the Euclidean center constrained to lie inside the polygon as well as the Euclidean center constrained to lie on the boundary of the polygon need not be unique.

A. Center Constrained to a Polygonal Region

We solve a slightly more general problem than the one mentioned in the introduction. Suppose we are given a set $S = \{s_1, s_2, \ldots, s_k\}$ of k points (in general position) in the plane E^2, and an n vertex simple polygon P. We wish to find the point c in P whose maximum distance to any point in S is minimized. If c is not constrained to lie in P, then it is the Euclidean center of S. However, we refer to c as the Euclidean center of S constrained to P and denote it by $EC_P(S)$.

Our algorithms make use of the *furthest point Voronoi diagram* of the set S, denoted as $FPVD(S)$. Given a point $x \in E^2$, we let $\phi(x)$ denote the furthest neighbors of x in S, that is the set of points in S such that $d(x, \phi(x)) = \max_{y \in S} d(x, y)$ where d is the Euclidean distance function. The $FPVD(S)$ partitions the plane into unbounded convex cells, $V(s_i)$, such that for any point $p \in V(s_i)$, $s_i \in \phi(p)$. This structure can be computed in $O(n \log n)$ time [21]. A list of the many geometric properties of the furthest point Voronoi diagram can be found in [18], [21], [6].

Notice that if the Euclidean center happens to lie inside the constraining polygon, then it is also the constrained Euclidean center. However, difficulties arise when the Euclidean center does not lie inside the polygon. These difficulties are resolved in the following lemmas.

Lemma II.1 *The Euclidean center of S constrained to lie in P is the midpoint of $DIAM(S)$ provided that the diametral circle contains the set S, and the midpoint is contained in P.*

Before tackling the problem of determining the location of $EC_P(S)$ when it is not on the midpoint of $DIAM(S)$, we first establish a lemma that will prove useful. Let a, b be two points in S such that both a and b are on the convex hull of S, $[ab]$ is not the diameter of S, and $V(a)$ and $V(b)$, the two cells of $FPVD(S)$ representing a and b, respectively, are adjacent and separated by an edge e. Let x be a point on the interior of e, and let $\epsilon > 0$ be any small constant.

Lemma II.2 *There exists a point $y \in e$ with $d(x, y) < \epsilon$ such that $d(y, a) < d(x, a)$ and $d(y, b) < d(x, b)$.*

Proof: The edge e must lie on the bisector of line segment $[ab]$, since the points on e are equidistant from both a and b. The points a, b, x must form a triangle because otherwise $[ab]$ would be the diameter. Since x is contained in $int(e)$, let y be a point on e in $\triangle(abx)$ such that $d(x, y) < \epsilon$. The lemma follows. ■

We now complete the characterization of $EC_P(S)$.

Lemma II.3 *If the Euclidean center of S constrained to lie in P is not the midpoint of $DIAM(S)$, then it lies on one of the following points that yields the smallest spanning circle:*

1. *a vertex of the $FPVD(S)$ contained in P,*
2. *a proper intersection point of the $FPVD(S)$ and the boundary P,*
3. *a vertex of the polygon P,*
4. *a point x on an edge e of P with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d(y, \phi(x)) \geq d(x, \phi(x))$.*

Proof: If $EC_P(S)$ does not lie on any of the points mentioned in the statement of the lemma, then it must lie in one of the regions described in the following four cases. We show that each of these cases leads to a contradiction. For simplicity of exposition, let $c = EC_P(S)$.

Case 1: c is a point in the interior of a cell of the $FPVD(S)$, and in $int(P)$. Let $V(b)$ be the cell containing c. By the Jordan Curve Theorem [19], line segment $[bc]$ must intersect ∂P or $V(b)$ since $b \notin V(b)$. Let x be the intersection point closest to c. The point x must be in $V(b)$. Therefore the circle centered at x with radius $d(x, b)$ encloses the set S. However, $d(x, b) < d(c, b)$ by construction. Hence, we have a contradiction.

Case 2: c is a point in the interior of a cell of the $FPVD(S)$, and in the interior of an edge e of P but does not satisfy the property that $\forall y \in e$, if $\phi(y) = \phi(c)$ then $d(y, \phi(c)) \geq d(c, \phi(c))$. Since the latter property is not satisfied, a point $x \in e$ such that $\phi(x) = \phi(c)$ and $d(x, \phi(c)) < d(c, \phi(c))$ must exist. However, the very existence of x contradicts that c is the constrained Euclidean center since the circle centered at x with radius $d(x, \phi(c))$ encloses S.

Case 3: c is a point in the interior of an edge e of the $FPVD(S)$, and in $int(P)$. Let $V(a)$ and $V(b)$ be the two cells separated by the edge e. Since c is not on the diameter of the set S, by Lemma II.2 we know that there exists a point x in e and in $int(P)$ such that $d(x, a) < d(c, a)$ and $d(x, b) < d(c, b)$. This contradicts that c is the constrained Euclidean center.

Case 4: c is a point in the interior of an edge e_v of the $FPVD(S)$, and in the interior of an edge e_p of P such that e_v and e_p intersect but not properly. Same argument as Case 3.

■

Lemma II.1 and Lemma II.3 characterize the location of $EC_P(S)$. We outline the following algorithm to compute this point.

Algorithm 1: *Euclidean Center of P constrained to lie in S*

Input: A set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a simple polygon $P = \{p_1, p_2, \ldots, p_n\}$.

Output: $EC_P(S)$

1. Compute the $FPVD(S)$.
2. Compute $DIAM(S)$.
3. Compute the circle C having $DIAM(S)$ as diameter.
4. Preprocess P in $O(n \log n)$ time for point inclusion testing in $O(\log n)$ time using the algorithms of Kirkpatrick [15] or Sarnak and Tarjan [23].
5. If the midpoint of C is contained in P and all the points of S are contained in C then exit with the midpoint of $DIAM(S)$.
6. Compute the set of vertices of $FPVD(S)$ contained in P. Let V_c represent this set.
7. Compute the set of intersections $I_c = \{i_1, i_2, \ldots, i_k\}$ of P with $FPVD(S)$.
8. Partition each edge e_i of P such that for every pair of points $x, y \in e_i$, we have that $\phi(x) = \phi(y)$. Denote the j^{th} partition of e_i by e_{ij}.
9. For each e_{ij}, compute the point on e_{ij} closest to $\phi(e_{ij})$. If this point is not an endpoint of e_{ij}, place it in the set E_c.
10. Let P_c represent the vertices of P. For each point c in V_c, I_c, P_c, E_c, compute the smallest spanning circle with center c. Let SP represent this set.
11. Select all the smallest circles in SP, and output their centers and the radius.

Notice that we assumed that the number of vertices of P equals the number of points in S. Clearly, this need not be the case, however, this assumption simplifies the complexity of notation. It is straightforward to repeat the complexity analysis when P and S have different cardinalities.

Theorem II.1 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a simple polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie in P in time $O(n \log n + k)$ where n is the size of the input and k is the number of intersections between the edges of the $FPVD(S)$ and P.*

For simple polygons, k can be $O(n^2)$, however, for convex polygons, we notice the following: a line segment can intersect a convex polygon only twice. Therefore, since $FPVD(S)$ consists of $O(n)$ line segments, there can only be $O(n)$ intersections between $FPVD(S)$ and an n vertex convex polygon. Therefore, we have:

Corollary II.1 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a convex polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie in P in time $O(n \log n)$ where n is the size of the input.*

B. Center Constrained to a Polygonal Chain

With a slight modification, Algorithm 1 can compute the Euclidean center constrained to lie on the boundary of the polygon, denoted as $EC_{\partial P}(S)$. These modifications are outlined below.

Lemma II.4 *The Euclidean center of S constrained to lie on the boundary of P is the midpoint of $DIAM(S)$ provided that the diametral circle contains the set S, and the midpoint is on the boundary of P.*

Lemma II.5 *If the Euclidean center of S constrained to lie on the boundary of P is not the midpoint of $DIAM(S)$, then it lies on one of the following points that yields the smallest spanning circle:*

1. *a vertex of the $FPVD(S)$ on the boundary of P,*
2. *an intersection point of the $FPVD(S)$ and the boundary P,*
3. *a vertex of the polygon P,*
4. *a point x on an edge e of P with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d(y, \phi(x)) \geq d(x, \phi(x))$.*

Lemma II.4 and Lemma II.5 characterize the location of $EC_{\partial P}(S)$. The modifications to Algorithm 1 for computing these points are straightforward. Therefore, we conclude with the following.

Theorem II.2 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a simple polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie on the boundary of P in time $O(n \log n + k)$ where n is the size of the input and k is the number of intersections between the edges of the $FPVD(S)$ and P.*

Corollary II.2 *Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$ and a convex polygon $P = \{p_1, p_2, \ldots, p_n\}$, we can compute the Euclidean center of S constrained to lie on the boundary P in time $O(n \log n)$ where n is the size of the input.*

III. Constrained Geodesic Center

Both versions of the constrained Euclidean center serve as good first approximations for the locations of the pin gate. However, in some cases the constrained Euclidean center may not be a point satisfying Property 1, as intended. In fact, it may be quite bad in the sense that the liquid may have to travel quite far despite the fact that the pin gate is located at the constrained or boundary-constrained Euclidean center. The reason is that the Euclidean distance of the pin gate to all the points may not be a good measure of the actual distance the liquid must travel inside the polygon. For example, in Figure 2, the Euclidean center, constrained Euclidean center and boundary-constrained Euclidean center all lie on the same vertex indicated on the polygon. However, the distance that the liquid must travel inside the polygon from that point to vertex v is quite large compared to vertex c. Although for convex or "near" convex objects, the Euclidean metric may be good, it seems that the geodesic metric may serve as a better approximation since liquid is traveling inside the polygon.

In the geodesic metric, the distance between two points inside a simple polygon is defined as the length of the shortest path connecting the two points inside the polygon. The *geodesic center* of a simple polygon is the point whose maximum geodesic distance to any other point in the polygon is minimized. Therefore, by definition, the geodesic center

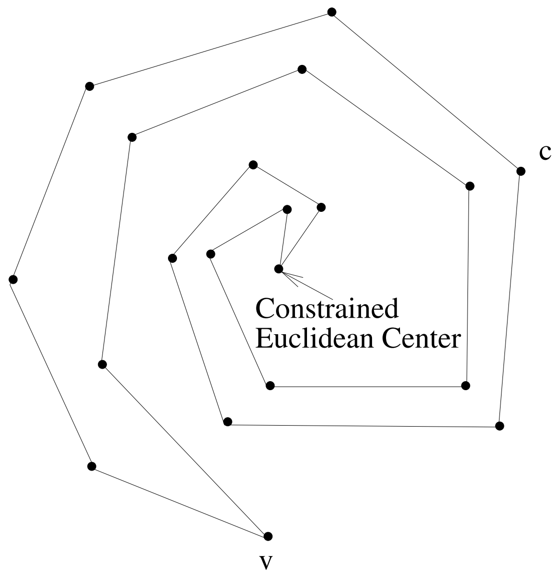

Fig. 2. The constrained Euclidean center may not be a good approximation for the best pin gate location.

of a simple polygon lies inside the polygon. Although the geodesic center and boundary-constrained geodesic center may serve as better approximations for the location of a pin gate, computing both centers is more difficult than their Euclidean counter-parts as we shall see.

A. Geometric Properties

The problem of computing the geodesic center of a simple n vertex polygon P, denoted $GC(P)$, was first tackled by Asano and Toussaint [4]. They gave an $O(n^3 \log \log n)$ time algorithm for computing the center. In [4], it is shown that the geodesic center is unique and located on a vertex of the *geodesic furthest point Voronoi diagram* of P, denoted $GFPVD(P)$. The $GFPVD(P)$, like its Euclidean counterpart, divides the polygon P into cells $V(v_i)$, such that the locus of points in $V(v_i)$ is further from v_i than any other vertex of P (with distance measured with the geodesic metric). Later, Pollack, Rote and Sharir [20] reduced the complexity of computing the geodesic center to $O(n \log n)$ time. They used a different approach and achieved their time bound by a modification of Meggido's technique. Recently, Aronov et al.[2] presented an $O(n \log n)$ time algorithm for computing the $GFPVD(P)$, thus providing an alternate $O(n \log n)$ time solution for computing the center. Therefore, to compute the geodesic center of a simple polygon, any one of the above algorithms may be used, however, all of these algorithms are complicated.

The problem of computing the boundary-constrained geodesic center of a simple polygon P, denoted as $GC(\partial P)$, has not previously been addressed. We concentrate on solving this problem. Like its Euclidean counter-part, the geodesic center constrained to the boundary is not necessarily unique, and not necessarily an intersection point of $GFPVD(P)$ and P. If an algorithm for computing the geodesic center already exists, a good heuristic for computing the boundary-constrained geodesic center is to compute the point on the boundary closest to the geodesic center. In some cases, this heuristic actually gives the boundary-constrained geodesic center. In the next section, we present an $O(n \log n)$ time algorithm to compute the boundary-constrained geodesic center exactly. The main idea behind the algorithm is the following. We divide the polygon boundary into polygonal chains such that the geodesic furthest neighbor of any point on a given chain is the same. Then, we compute for each chain, the point, which we call the candidate for that chain, whose distance to the furthest neighbor is the smallest compared to any other point on the chain. We select the smallest candidates as the geodesic center constrained to the boundary. We modify an algorithm of Suri [25], similar to [2], to compute this.

Given two points a, b in a polygon P, there is a unique geodesic path connecting a, b in P. We denote this path by $\pi(a, b)$ and its length by $d_G(a, b)$. Since geodesic distance is a metric, the triangle inequality holds. Therefore, we have that $d_G(x, y) \leq d_G(x, z) + d_G(z, y)$ for every three points x, y, z in P. The *geodesic furthest neighbors* of a point x in P, denoted by $\phi(x)$, are the set of points y in P such that $d_G(x, y) = \max_{\forall\ z \in P}\{d_G(x, z)\}$. Asano and Toussaint [4] showed that the geodesic furthest neighbor of a point is always a convex vertex of the polygon. The *geodesic diameter* of a polygon P, denoted as $GDIAM(P)$, is determined by the pair of points in P whose geodesic distance is maximum over all pairs of points in P. If two shortest paths do not share a point, we say they are *disjoint*; otherwise, we say that the paths *intersect*.

An important property of geodesics, at the heart of the algorithm, is the *Crossing Property* described in the following lemma.

Lemma III.1 (Crossing Property) *[25] Let p_1, p_2, p_3, p_4 be four points in this order on the boundary of P. Suppose that $p_3 \in \phi(p_2)$ and $p_4 \in \phi(p_1)$. Then we also have $p_3 \in \phi(p_1)$ and $p_4 \in \phi(p_2)$.*

To compute the boundary-constrained geodesic center, we first compute a constrained geodesic decomposition of the boundary of polygon P, which is a decomposition of the boundary of P into polygonal chains $(c_1, c_2, \ldots, c_t)$ such that $\bigcup_{i=1}^{t} c_i = P$ and for every $x, y \in c_i$, $\phi(x) = \phi(y)$. We denote this decomposition as ∂-$CGD(P)$. Given this decomposition, the constrained geodesic center can be easily identified, as shall be shown in the next section. The crossing property is the key behind the algorithm. It suggests a divide-and-conquer approach to solving the problem of computing the constrained geodesic decomposition of ∂P. We first consider a restricted version of the decomposition problem, and then we show how to use its solution to compute the whole decomposition.

B. Restricted Geodesic Decomposition

The restricted version of the decomposition problem is described as follows. Let $U = (u_a, \ldots, u_b)$ be the counterclockwise chain from point u_a to point u_b on the boundary of P. Let $V = (v_c, \ldots, v_d)$ be the clockwise chain from v_c to v_d on the boundary of P, such that both chains are disjoint except possibly at the endpoints. The set of points which are the furthest neighbors of x restricted to V is denoted by $\phi_V(x)$. We want to decompose U into polygonal chains $(c_1', c_2', \ldots, c_s')$ such that $\bigcup_{i=1}^{s} c_i' = U$ and for every $x, y \in c_i'$, $\phi_V(x) = \phi_V(y)$. We refer to this decomposition as the restricted decomposition of U with respect to V, denoted by $RGD_V(U)$.

Let u_a, u_b, v_d, v_c be four points on ∂P appearing in that order in a counterclockwise traversal of ∂P. $P[u_a, u_b; v_c, v_d]$ denotes the region of P obtained by joining the counterclockwise chain of ∂P from u_a to u_b and the clockwise chain of ∂P from v_c to v_d with $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$. We say that a polygonal region $R \subset P$ is *geodesically convex* if for every pair of points $x, y \in R$, we have that $\pi(x, y) \in R$.

Lemma III.2 *[25] $P[u_a, u_b; v_c, v_d]$ is geodesically convex.*

Lemma III.2 implies that the restricted geodesic decomposition of U with respect to chain V can be done entirely within $P[u_a, u_b; v_c, v_d]$. Below, we outline the algorithm to compute this decomposition.

Algorithm 2: $RGD(P[u_a, u_b; v_c, v_d])$

1. If $P[u_a, u_b; v_c, v_d]$ is degenerate then
   (a) Find $v_m \in [v_c, \ldots, v_d]$ such that $d_G(u_a, v_m) = \max\{d_G(u_a, v_j) | c \leq j \leq d\}$;
   (b) Find the point $u_{\min} \in [u_a, \ldots, u_b]$ such that $d_G(u_{\min}, v_m) = \min\{d_G(u_i, v_m) | a \leq i \leq b\}$.
   (c) Output $u_{\min}$, v_m, $[u_a, \ldots, u_b]$, $d_G(u_{\min}, v_m)$.
2. Else if $(b - a) \leq 2$ and u_a, u_b are vertices then
   (a) Determine $\phi(u_a)$ and $\phi(u_b)$.
   (b) If $\phi(u_a) = \phi(u_b)$ then
      i. let $v_m = \phi(u_a)$.
      ii. Find the point $u_{\min} \in [u_a, u_b]$ such that $d_G(u_{\min}, v_m) = \min\{d_G(p, v_m) | p \in [u_a, u_b]\}$.
      iii. Output $u_{\min}$, v_m, $[u_a, u_b], d_G(u_{\min}, v_m)$.
   (c) Else
      i. Compute and sort the partition points $x_1, x_2, \ldots, x_s$ on edge $[u_a u_b]$. Let $[u_1, u_2, \ldots, u_{s+2}]$ be the points in sorted ordered on edge $[u_a, u_b]$.
      ii. let $k = \lceil (s + 2)/2 \rceil$.
      iii. Find a $v_m \in [v_c, \ldots, v_d]$ such that $d_G(u_k, v_m) = \max\{d_G(u_k, v_j) | c \leq j \leq d\}$
      iv. Call $RGD(P[u_a, u_k; v_m, v_d])$ and $RGD(P[u_k, u_b; v_c, v_m])$.
3. Else if $(b - a) \leq 2$ and u_a, u_b are not both vertices then
   (a) Determine $\phi(u_a)$ and $\phi(u_b)$.
   (b) If $\phi(u_a) = \phi(u_b)$ then
      i. let $v_m = \phi(u_a)$.

ii. Find the point $u_{\min} \in [u_a, u_b]$ such that $d_G(u_{\min}, v_m) = \min\{d_G(p, v_m) | p \in [u_a, u_b]\}$.

iii. Output $u_{\min}$, v_m, $[u_a, u_b], d_G(u_{\min}, v_m)$.

(c) Else

i. solve directly.

4. Else

(a) let $k = \lceil (a+b)/2 \rceil$

(b) Find a $v_m \in [v_c, \dots, v_d]$ such that $d_G(u_k, v_m) = \max\{d_G(u_k, v_j) | c \leq j \leq d\}$

(c) Call $RGD(P[u_a, u_k; v_m, v_d])$ and $RGD(P[u_k, u_b; v_c, v_m])$.

Not only does algorithm *RGD* compute the restricted geodesic decomposition of U into polygonal chains $(c_1', c_2', \dots, c_s')$ such that $\bigcup_{i=1}^{s} c_i' = U$ and for every $x, y \in c_i'$, $\phi_V(x) = \phi_V(y)$, but for each chain c_i' it computes the point on the chain whose distance to its furthest neighbor is minimum. We prove the correctness of the algorithm and at the same time, elaborate on some of the steps such as how to compute the partition in Step 2.

We say that $P[u_a, u_b; v_c, v_d]$ is *degenerate* if $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ are not disjoint. Given a degenerate instance of $P[u_a, u_b; v_c, v_d]$ computing the decomposition of U with respect to V is straightforward. Let x be a point on $\pi(u_a, v_c) \cap \pi(u_b, v_d)$. Every shortest path between a point y in U and a point z in V must contain x. Therefore, the point v_f of V furthest from x is also the point furthest from all points in U. The point v_f can be computed by traversing the shortest path tree of x. Since this tree can be computed in linear time [14], we conclude with the following.

Lemma III.3 *If $P[u_a, u_b; v_c, v_d]$ is degenerate and v_f is the furthest point from $x \in \pi(u_a, v_c) \cap \pi(u_b, v_d)$, then the point v_f on V is $\phi_V(z)$ for all $z \in U$. The point v_f can be computed in time proportional to the size of $P[u_a, u_b; v_c, v_d]$.*

Given an instance of $P[u_a, u_b; v_c, v_d]$, if $(u_a, \dots, u_b)$ is a polygonal chain, then by the crossing property, we can divide the chain in half and recurse. However, if u_a and u_b are the endpoints of an edge, it is not clear how to proceed. In such a situation, we resolve the problem by partitioning the edge into subedges. We require the following property on each subedge s_i of $[u_a, u_b]$. For every pair of points x, y on s_i, we want the shortest path from x to every vertex v_j on $(v_c, \dots, v_d)$ to be identical to the shortest path from y to v_j except for the first link. We refer to this property as the *path-invariant* property of a subedge. This partitioning scheme can be easily computed given the shortest path tree of u_a and u_b.

Let m be the size of $P[u_a, u_b; v_c, v_d]$. Since $P[u_a, u_b; v_c, v_d]$ is triangulated, the shortest path tree of u_a and u_b can be obtained in $O(m)$ time using the algorithm of [14]. Once the shortest path trees have been computed, all the partition points on the edge can also be obtained in O(m) time, by traversing the two trees. Finally, $O(m \log m)$ time is used to sort the partition points. Hence, we conclude with the following.

Lemma III.4 *Given an instance of $P[u_a, u_b; v_c, v_d]$ of size m, where $[u_a, u_b]$ is an edge, we can partition in $O(m \log m)$ time the edge $[u_a, u_b]$ into subedges such that each subedge respects the path-invariant property with respect to the chain $(v_c, \dots, v_d)$.*

The main reason we partition the edge into subedges, when faced with an instance of $P[u_a, u_b; v_c, v_d]$ where $[u_a, u_b]$ is an edge, is that the base problem that we are faced with at the end of the recursion can be solved directly due to the path-invariant property. As the algorithm computes the decomposition of the U chain, eventually in Step 3, we are faced with an instance $P[u_a, u_b; v_c, v_d]$ where u_a, u_b are the endpoints of a subedge respecting the path-invariant property, and $(v_c, \dots, v_d)$ is a polygonal chain.

We need to decompose $[u_a, u_b]$ into pieces such that in each piece the farthest geodesic neighbor is the same. Let $d_{v_i}(x)$ denote the distance function from $v_i \in (v_c, \dots, v_d)$ to a point x in $[u_a, u_b]$. These functions are simple and can be used to solve directly the decomposition of $[u_a, u_b]$ by computing the upper envelope of the functions $d_{v_i}(x)$ for all $v_i \in (v_c, \dots, v_d)$.

Lemma III.5 *Let p_1, p_2, p_3, p_4 be four points in this order on the boundary of P. If $d_G(p_1, p_4) > d_G(p_1, p_3)$ then $d_G(p_2, p_4) > d_G(p_2, p_3)$.*

The above lemma implies that we can compute the upper envelope in linear time simply by inserting the functions in the order $(v_c, \dots, v_d)$ or the reverse order.

We conclude with the following.

Lemma III.6 *Given an instance of $P[u_a, u_b; v_c, v_d]$ where $[u_a, u_b]$ is a segment with the path-invariant property, $[u_a, u_b]$ can be decomposed, in time proportional to the size of $P[u_a, u_b; v_c, v_d]$, into subedges such that for each point in the subedge, the furthest neighbor is the same vertex of $(v_c, \dots, v_d)$.*

The algorithm to compute $RGD_V(U)$ stems from the crossing property described in Lemma III.1. This property holds at all levels of recursion.

Lemma III.7 *Let $U = (u_a, \dots, u_b)$ and $V = (v_c, \dots, v_d)$. Given an initial call of $RGD(P[u_a, u_b; v_c, v_d])$, every recursive call $RGD(P[u_q, u_r; v_s, v_t])$ has the property that for all $x \in (u_q, \dots, u_r)$, we have that $\phi_V(x) \in (v_s, \dots, v_t)$.*

The correctness of algorithm *RGD* follows from Lemmas III.7, III.3, III.6, and the crossing property.

Theorem III.1 *Algorithm RGD correctly computes the restricted geodesic decomposition of chain U with respect to V.*

We now turn our attention to the complexity analysis of algorithm *RGD*. We show that the algorithm runs in $O(n \log n)$ time and uses $O(n)$ space. To do this, we show that there are $O(\log n)$ levels of recursion. Then we show that an instance of $RGD(P[u_a, u_b; v_c, v_d])$ (excluding recursive calls and sorting of partition points) runs in time proportional to the size of $P[u_a, u_b; v_c, v_d]$. Finally, we show that the total size of all polygons at a particular level of recursion is $O(n)$. The main ideas in the complexity analysis to follow stem from the analysis given in Suri[25].

Lemma III.8 *Algorithm $RGD(P[u_a, u_b; v_c, v_d])$ runs in time proportional to the size of $P[u_a, u_b; v_c, v_d]$, excluding recursive calls and sorting partition points.*

The number of *distinct* edges among all the polygons constructed by algorithm RGD is $O(n)$, where n is the size of the polygon in the initial invocation. Recall that $P[u_a, u_b; v_c, v_d]$ denotes the region of P obtained by joining the counterclockwise chain of ∂P from u_a to u_b and the clockwise chain of ∂P from v_c to v_d with $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ We refer to $\pi(u_a, v_c)$ and $\pi(u_b, v_d)$ as *connecting paths*. There are three types of edges in $P[u_a, u_b; v_c, v_d]$. An edge that belongs to ∂P is a *primary edge*, an edge that is a subedge of an edge belonging to ∂P is a *partition edge*, and an edge that belongs to a connecting path is a *connecting edge*. The number of distinct primary edges is $O(n)$ since all primary edges are contained in the initial polygon. Also, there are $O(n)$ distinct partition edges and connecting edges.

Lemma III.9 *The total number of partition points added is $O(n)$ where n is the size of the initial instance of $RGD(P[u_a, u_b; v_c, v_d])$.*

Since a total of $O(n)$ partition points are added given that the size of the initial instance of $RGD(P[u_a, u_b; v_c, v_d])$ is n, we conclude that the total time spent sorting all partition points is $O(n \log n)$. Therefore we have the following.

Lemma III.10 *The total time spent sorting all partition points is $O(n \log n)$ for an initial instance of $RGD(P[u_a, u_b; v_c, v_d])$ of size n.*

An edge is partitioned at most once, and at each level of recursion, the U chain is divided in half. Thus, we see that there can only be $O(\log n)$ levels of recursion.

Lemma III.11 *There are $O(\log n)$ levels of recursion where n is the size of the initial instance of $RGD(P[u_a, u_b; v_c, v_d])$.*

All that remains to be seen is that the summed complexity of all the polygons constructed in one particular level of recursion is $O(n)$. A distinct edge can belong to only a constant number of polygons in a particular level of recursion. By the construction of polygons in Step 2 and Step 4, we see that partition edges and primary edges cannot occur in two polygons at the same level of recursion. This follows from the way the U chain and V chain are divided. Finally, a connecting edge can only occur in a constant number of polygons on the same level of recursion. Therefore, we conclude with the following theorem

Theorem III.2 *Algorithm RGD computes the restricted geodesic decomposition of chain U with respect to V using $O(n \log n)$ time and $O(n)$ space given an input of size n.*

In the next section, we show how to use the restricted geodesic decomposition to solve our initial problem of computing the geodesic decomposition of the boundary.

C. Geodesic Center Constrained to the Boundary

To compute the geodesic decomposition of the boundary of a simple polygon, we apply the algorithm for restricted decomposition three times. The following lemma of Suri[25] provides the key.

We assume that $(u_1, u_2, \ldots, u_n)$ is the counterclockwise sequence of vertices of polygon P in the lemma to follow. Let $(u_a, \ldots, u_b)$ denote the counterclockwise chain of ∂P from u_a to u_b. Let u_i be an arbitrary vertex of P. Let $u_j \in \phi(u_i)$ be a geodesic furthest neighbor of u_i, and $u_k \in \phi(u_j)$ be a geodesic furthest neighbor of u_j. It is possible that $u_i = u_k$. Let us assume, without loss of generality, that u_i, u_j, u_k is the order of these vertices in a counterclockwise traversal of P starting at vertex u_i, then we have the following lemma.

Lemma III.12 *[25] Let u_i be an arbitrary vertex of P. Let $u_j \in \phi(u_i)$ and let $u_k \in \phi(u_j)$, such that u_i, u_j, and u_k are in this order in a counterclockwise traversal of P.*

1. *for any vertex $u_l \in (u_i, \ldots, u_j)$, there exists another vertex $u_m \in (u_j, \ldots, u_i)$ satisfying $u_m \in \phi(u_l)$,*
2. *for any vertex $u_l \in (u_j, \ldots, u_k)$, there exists another vertex $u_m \in (u_k, \ldots, u_i, \ldots, u_j)$ satisfying $u_m \in \phi(u_l)$,*
3. *for any vertex $u_l \in (u_k, \ldots, u_i)$, there exists another vertex $u_m \in (u_i, \ldots, u_j, \ldots, u_k)$ satisfying $u_m \in \phi(u_l)$,*

From the above lemma, we can conclude that to compute the geodesic decomposition of P, we simply solve the following three instances of the restricted geodesic decomposition of P.

Instance 1 The U chain is $(u_i, \ldots, u_j)$ and the V chain is $(u_j, \ldots, u_k, \ldots, u_i)$.
Instance 2 The U chain is $(u_j, \ldots, u_k)$ and the V chain is $(u_k, \ldots, u_i, \ldots, u_j)$.
Instance 3 The U chain is $(u_k, \ldots, u_i)$ and the V chain is $(u_i, \ldots, u_j, \ldots, u_k)$.
Therefore, we have the following theorem.

Theorem III.3 *The geodesic decomposition of a simple polygon can be computed in $O(n \log n)$ time and $O(n)$ space given an input of size n.*

Once the geodesic decomposition of a polygon P has been computed, the boundary-constrained geodesic center can be computed as follows. Let $(c_1, c_2, \ldots, c_t)$ represent the polygonal chains in the geodesic decomposition of the boundary of P where $\bigcup_{i=1}^{t} c_i = P$ and for every $x, y \in c_i$, $\phi(x) = \phi(y)$. For each c_i, compute the point $x \in c_i$, with the property that the geodesic distance from x to $\phi(x)$ is smallest compared to all other points in c_i. In other words, $d_G(x, \phi(x)) = \min_{\forall y \in c_i}\{d_G(y, \phi(y))\}$. The point x is referred to as the *candidate* for the chain c_i. In fact, algorithm RGD already computes the candidates for each chain as seen in steps 1, 2 and 3 of the algorithm. We conclude with the following theorem.

Theorem III.4 *The boundary-constrained geodesic center of polygon P is the candidate x^*, such that $d_G(x^*, \phi(x^*)) = \min_{\forall \text{ candidates } y}\{d_G(y, \phi(y))\}$.*

D. Geodesic Center Constrained to a Polygonal Region

In this section, we address the problem of computing the geodesic center of a simple polygon P constrained to lie inside a simple polygon Q, where Q is contained in P. We denote this center as $GC_Q(P)$. If Q equals P then we simply have the geodesic center of the polygon P. We can further restrict the geodesic center to lie on the boundary of polygon Q, denoted $GC_{\partial Q}(P)$. In this case, if Q equals P, then we have the geodesic center constrained to the boundary of P. The reason we differentiated the problem of computing the geodesic center constrained to the boundary from this problem is that we use the geodesic furthest point Voronoi diagram to solve this problem, but to solve the former problem, we were able to avoid computing the geodesic furthest point Voronoi diagram by modifying Suri's algorithm [25]. The arguments we use to solve this problem are similar to the arguments used to solve the Euclidean center constrained to a polygon region.

Since in this and the following subsection we make extensive use of the $GFPVD(P)$, let us review a few of its properties. In order to use the algorithm of [2], we assume that no vertex is geodesically equidistant from two other vertices. This can always be guaranteed by applying a slight perturbation to the vertices if the condition is violated. Like its Euclidean counter-part, $GFPVD(P)$ partitions P into cells, $V(p_i)$, such that for every point $p \in V(p_i)$, the point p_i is a furthest geodesic neighbor of p. A vertex of the $GFPVD(P)$ is a point that is geodesically equidistant to three vertices furthest from it. An edge between two Voronoi vertices is either a straight edge or a hyperbolic arc. Finally, the boundary of a Voronoi cell consists of a concatenation of straight edges and hyperbolic arcs. For more geometric properties of geodesic furthest point Voronoi diagrams, the reader is referred to [2], [3].

Lemma III.13 *[3], [20] The geodesic center of a simple polygon P lies on the midpoint of the geodesic diameter of P ($GDIAM(P)$) or on a vertex of the $GFPVD(P)$.*

When the geodesic center of the polygon P lies on the midpoint of the geodesic diameter, it has a special property. Let $bis(a, b)$ represent the geodesic bisector of a and b inside P, i.e. for every point x on $bis(a, b)$, $d_G(x, a) = d_G(x, b)$. Let a, b be two points of polygon P, then we have the following.

Lemma III.14 *If the midpoint m of $\pi(a, b)$ lies on the interior of the edge separating cells $V(a)$ and $V(b)$ of $GFPVD(P)$, then m is the geodesic center P and $\pi(a, b)$ is the geodesic diameter of P.*

Before continuing, we need a few definitions. Let a, b, c be three points in a simple polygon P. The geodesic angle $\angle abc$ is the smaller of the two angles between the first link on the geodesic path from b to a and the first link on the path from b to c. Now, consider the paths $\pi(a, b), \pi(b, c)$, and $\pi(a, c)$. There exist points a', b', and c' such that the paths $\pi(a, b)$ and $\pi(a, c)$ intersect in the path $\pi(a, a')$, the paths $\pi(b, c)$ and $\pi(b, a)$ intersect in the path $\pi(b, b')$, and the paths $\pi(c, a)$ and $\pi(c, b)$ intersect in the path $\pi(c, c')$. The three paths $\pi(a', b'), \pi(b', c')$, and $\pi(c', a')$ form what is known as a *geodesic triangle*, denoted $\triangle a'b'c'$. The vertices a', b', c', are the only convex vertices of the geodesic triangle and are referred to as the *peaks* of the triangle. Pollack et al.[20] proved the following property concerning geodesic triangles.

Lemma III.15 *[20] If the geodesic angle $\angle ba'c$ at a' is greater than or equal to $\pi/2$, then $d_G(b, c) > d_G(a', b), d_G(a', c)$*

Lemma III.16 *The geodesic center of P constrained to lie in Q is the midpoint m of $GDIAM(P)$ provided that m is the geodesic center of P and lies in Q.*

To address the problem of determining the location of $GC_Q(P)$ when it does not satisfy the conditions of the above lemma, we establish the following lemmas. Let a, b be two vertices of P such that they each have a corresponding cell $V(a)$ and $V(b)$, respectively, which are adjacent separated by an edge e in $GFPVD(P)$. Also, $\pi(a, b)$ is not the geodesic diameter of P. Let x be a point on the interior of e, and let $\epsilon > 0$ be any small constant.

Lemma III.17 *There exists a point $y \in e$ with $d_G(x, y) < \epsilon$ such that $d_G(y, a) < d_G(x, a)$ and $d_G(y, b) < d_G(x, b)$.*

Lemma III.18 *A point b of P cannot lie in $V(b)$.*

We now complete the characterization of $GC_Q(P)$.

Lemma III.19 *If the geodesic center of P constrained to lie in Q is not the midpoint of $GDIAM(P)$, then it lies on one of the following points:*

1. *a vertex of the $GFPVD(P)$ contained in Q,*
2. *a proper intersection point of the $GFPVD(P)$ and the boundary of Q,*
3. *a vertex of the polygon Q,*
4. *a point x on an edge e of Q with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d_G(y, \phi(x)) \geq d_G(x, \phi(x))$.*

We outline the algorithm to compute $GC_Q(P)$.

Algorithm 3: *Geodesic Center of P constrained to Q*

Input: A simple polygon $P = \{p_1, p_2, \ldots, p_n\}$, and a simple polygon $Q = \{q_1, q_2, \ldots, q_n\}$ with $Q \subset P$.

Output: $GC_Q(P)$

1. Compute the $GFPVD(P)$ using the algorithm of Aronov et al.[2].
2. Compute $GC(P)$ using the algorithm of Pollack et al.[20].
3. Preprocess Q in $O(n \log n)$ time for point inclusion testing in $O(\log n)$ time using the algorithms of Kirkpatrick [15] or Sarnak and Tarjan [23].

4. If $GC(P)$ is contained in Q then exit with $GC(P)$ as $GC_Q(P)$.
5. Preprocess P for shortest path queries using the algorithm of Guibas and Hershberger [13].
6. Compute the set of vertices of $GFPVD(P)$ contained in Q. Let V_c represent this set.
7. Compute the set of intersections $I_c = \{i_1, i_2, \ldots, i_k\}$ of Q with $GFPVD(P)$ using the algorithm of Chan[8].
8. Partition each edge e_i of Q such that for every pair of points $x, y \in e_i$, we have that $\phi(x) = \phi(y)$. Denote the j^{th} partition of e_i by e_{ij}.
9. For each e_{ij}, compute the point on e_{ij} closest to $\phi(e_{ij})$. If this point is not an endpoint of e_{ij}, put it in set E_c.
10. Let P_c represent the vertices of Q.
11. Let $CAN = V_c \cup I_c \cup P_c \cup E_c$.
12. Find the set of points $G = \{x \in CAN \mid d_G(x, \phi(x)) = \min_{y \in CAN} d_G(y, \phi(y))\}$
13. Output the set G.

Notice that we assumed that the number of vertices of Q equals the number of vertices of P. Clearly, this need not be the case, however, this assumption simplifys the complexity of notation. It is quite straightforward to repeat the complexity analysis when P and Q have different cardinalities.

Theorem III.5 *Given a polygon $P = \{p_1, p_2, \ldots, p_n\}$ and a polygon $Q = \{q_1, q_2, \ldots, q_n\}$ contained in P, we can compute the geodesic center of P constrained to lie in Q in time $O(n(n+k))$ where n is the size of the input and k is the number of intersections between the edges of the $GFPVD(P)$ and Q.*

E. Geodesic Center Constrained to a Polygonal Chain

With a slight modification, Algorithm 3 can compute the geodesic center of P constrained to lie on the boundary Q, $GC_{\partial Q}(P)$. These modifications are outlined below.

Lemma III.20 *The geodesic center of P constrained to lie on the boundary of Q is the midpoint m of $GDIAM(S)$ provided that m is the geodesic center of P and lies on the boundary of Q.*

Lemma III.21 *If the geodesic center of P constrained to lie on the boundary of Q is not the midpoint of $GDIAM(P)$, then it lies on one of the following points:*

1. *a vertex of the $GFPVD(P)$ on the boundary of Q,*
2. *a proper intersection point of the $GFPVD(P)$ and the boundary of Q,*
3. *a vertex of the polygon Q,*
4. *a point x on an edge e of Q with the property that $\forall y \in e$, if $\phi(y) = \phi(x)$ then $d_G(y, \phi(x)) \geq d_G(x, \phi(x))$.*

Lemma III.20 and Lemma III.21 completely characterize the location of $GC_{\partial Q}(P)$. The modifications to Algorithm 3 for computing these points are straightforward. Therefore, we conclude with the following.

Theorem III.6 *Given a polygon $P = \{p_1, p_2, \ldots, p_n\}$ and a polygon $Q = \{q_1, q_2, \ldots, q_n\}$ contained in P, we can compute the geodesic center of P constrained to lie on the boundary of Q in time $O(n(n+k))$ where n is the size of the input and k is the number of intersections between the edges of the $GFPVD(P)$ and Q.*

IV. Constrained Link Center

In this section, we consider the second property attributed to a good pin gate location. Recall that the second property states that the maximum number of turns that the liquid takes on its path from the pin gate to any point in the object should be small. The link metric provides a geometric interpretation of this property. The link metric measures the number of turns or bends in a path between two points. We need a few definitions about link paths before continuing.

The *link distance* between two points x and y inside a polygon P, denoted $d_L(x, y)$, is the minimum number of edges in any polygonal path connecting x and y without intersecting the boundary of P. A path $\pi_L(x, y)$ between x and y is a *minimum link path* provided that the number of edges in $\pi_L(x, y)$ is equal to $d_L(x, y)$. The *k-neighborhood* or *k-disk* about a point $x \in P$ is defined as $N_k(x) = \{y \in P \mid d_L(x, y) \leq k\}$, and the *covering radius* $c(x)$ of x is the smallest k such that $P \subset N_k(x)$. The *link radius* is defined by $r_L(P) = \min_{x \in P} c(x)$ and the *link center* of P is defined by $LC(P) = \{x \in P \mid c(x) = r_L(P)\}$. In essence, the link center is the set of points in P whose maximum link distance to any point in P is minimized, precisely the set of potential pin gates satisfying the second property of a suitable pin gate.

We review the problem of computing the link center of a simple n vertex polygon P. The problem of computing the link center was first addressed by Lenhart et al.[17] who provided a simple $O(n^2)$ time algorithm to compute $LC(P)$. Note that the link center of P is not necessarily a point as is the case with the geodesic center of P, but the link center may in fact be a geodesically convex region contained in P. Later Djidjev et al.[10] reduced the time complexity of computing $LC(P)$ to $O(n \log n)$. Therefore, to compute the link center of a simple polygon, either of these two algorithms may be used.

The problem of computing the link center constrained to the boundary of a polygon P, denoted as $LC(\partial P)$, has not been addressed. In this section, we provide a simple algorithm to compute the set $LC^*(\partial P)$ which is a subset of $LC(\partial P)$. In some cases $LC^*(\partial P)$ is in fact be equivalent to $LC(\partial P)$.

A. Link Center Constrained to the Boundary

In this section, we provide a simple algorithm for computing $LC^*(\partial P)$, which is a subset of $LC(\partial P)$. The following very simple observations form the basis of the algorithm.

Observation IV.1 *If $LC(P) \cap \partial P$ is non-empty, then $LC(\partial P) = LC(P) \cap \partial P$.*

Observation IV.2 *If a point $z \notin LC(P)$ is visible from a point $x \in LC(P)$, then $c(z)$ is one greater than $c(x)$.*

Two points x, y in polygon P are said to be *visible* provided that the line segment $[x, y]$ is in P. Given a set of points X in polygon P, the *strong visibility set* of X in P is $\{z \in P \mid \forall\ x \in X, [xz] \in P\}$ and the *weak visibility set* of X in P is $\{z \in P \mid \exists\ x \in X, [xz] \in P\}$. If X happens to be a simple polygon inside P, Ghosh [12] has shown that the *weak visibility set* of X in P is also a simple polygon, referred to as the *weak visibility polygon* of P from X and denoted by $WVP(X, P)$. We now outline the algorithm.

Algorithm 4: *Compute $LC^*(\partial P)$*

1. Compute $LC(P)$.
2. Let $LC_1 = LC(P) \cap \partial P$. If LC_1 is non-empty, exit with LC_1.
3. Compute the weak visibility polygon of P from $LC(P)$.
4. Let $LC_2 = WVP(LC(P), P) \cap \partial P$. Exit with LC_2.

If LC_1 is non-empty, then $LC(\partial P)$ is equal to LC_1. If on the other hand, LC_1 is empty, then the set LC_2 must be a subset of $LC(\partial P)$ since the link center of the polygon is contained strictly in the interior of P and the covering radius of every point in LC_2 is one greater than the covering radius of a point in the link center. The complexity of the algorithm is dominated by Step 1 which can be computed in $O(n \log n)$ time using the algorithm of [10]. A simple modification to the algorithm in [10] is needed to compute the intersection of $LC(P)$ with the boundary of P in the same time complexity. Step 3 can be performed in $O(n)$ time using the algorithm of [12]. The parts of $WVP(LC(P), P)$ that are part of the boundary of P can be identified during the computation of the weak visibility polygon. Therefore, we conclude with the following theorem.

Theorem IV.1 *$LC^*(\partial P)$ can be computed in $O(n \log n)$ time.*

V. Discussion

Of the solutions presented in this paper, computing the Euclidean center, with or without constraints, as well as the link center, with or without constraints, are both conceptually and computationally simpler than computing the geodesic center. However, the Euclidean center may not always be a good candidate for the location of a pin gate as pointed out in Section III. The link center considered alone may also not be a suitable candidate since liquid inside a mold does not necessarily travel along a link path. Combining these two constraints may provide a better approximation (e.g. computing the Euclidean center constrained to lie in the link center).

The geodesic center, although computationally more expensive, seems to be a better measure in terms of the distance the liquid travels inside a mold. A combination of the link and geodesic centers may reap the benefits of both properties of an ideal pin gate location being satisfied. For example, computing the geodesic center constrained to lie in the link center may provide a better solution than considering the geodesic center by itself.

References

[1] Anselman, G.W., J. Cunningham, R.A. Green, J.C. Lee, E.H. Phelps, A.M. Prewitt, V. Rowell, L.E. Taylor, C.W. Ward, E.L. Kotzin (editors), *Analysis of Casting Defects*, American Foundrymen's Society, Des Plaines, Ill., 1974.

[2] Aronov. B., S. Fortune, G. Wilfong, The Furthest-Site Geodesic Voronoi Diagram. *Disc. & Comp. Geom.*, **9**, pp. 217-255, 1993.

[3] Asano, T. and G.T. Toussaint, Computing the Geodesic Center of a Simple Polygon, in *Perspectives in Computing: Discrete Algorithms and Complexity, Proceedings of Japan-US Joint Seminar*, Academic Press, Boston, pp. 65-79, 1986.

[4] Avis, D. and G. Toussaint, An Optimal Algorithm for Determining the Visibility of a Polygon from an Edge, *IEEE Trans. on Comp.,* **C-30-12**, pp. 910-914, 1981.

[5] Berk, A.A., *Computer Aided Design and Analysis for Engineers.* Blackwell Scientific Publications, Oxford, England, 1988.

[6] Bhattacharya, B., and G. Toussaint, On Geometric Algorithms that Use the Furthest-Point Voronoi Diagram, in *Computational Geometry*, North Holland, Amsterdam, pp. 43-62, 1985.

[7] Bose, P., Geometric and Computational Aspects of Manufacturing Processes, Ph.D. thesis, McGill University, 1994, also available as UBC tech. report 95-02, Univ. of British Columbia, 1995.

[8] Chan, T., A Simple Trapezoidal Sweep Algorithm for Reporting Red/Blue Segment Intersections, in *Proc. of the 6th Canadian Conf. on Comp. Geom.*, pp. 263-268, 1994.

[9] Chazelle, B., Triangulating a Simple Polygon in Linear Time, *Discrete and Computational Geometry*, **6**, pp. 485-524, 1991.

[10] Djidjev, H.N., A. Lingas, and J. Sack, An $O(n \log n)$ Algorithm for Computing the Link Center of a Simple Polygon. *Discrete and Computational Geometry*, **8**, pp. 131-152, 1992.

[11] ElWakil, S.D., *Processes and Design for Manufacturing*, Prentice Hall, New Jersey, 1989.

[12] Ghosh, S.K., Computing the Visibility Polygon from a Convex Set and Related Problems. *J. of Algorithms*, **12**, pp. 75-95, 1991.

[13] Guibas, L., and J. Hershberger, Optimal Shortest Path Queries in a Simple Polygon. *J. of Computer and System Sciences*, **39**, pp. 126-152, 1989.

[14] Guibas, L., J. Hershberger, D. Leven, M. Sharir, and R. Tarjan, Linear-Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons. *Algorithmica*, **2**, pp. 209-233, 1987.

[15] Kirkpatrick, D., Optimal Search in Planar Subdivisions, *SIAM J. of Computing*, **12**(1), pp. 28-35, 1983.

[16] Lyman, T. (editor), *Casting Design Handbook*, American Society for Metals, Ohio, 1962.

[17] Lenhart, W., R. Pollack, J. Sack, R. Seidel, M. Sharir, S. Suri, G. Toussaint, S. Whitesides, and C. Yap, Computing the Link Center of a Simple Polygon. *Discrete and Computational Geometry*, **3**, pp. 281-293, 1988.

[18] Okabe, A., B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, England, 1992.

[19] O'Rourke, J., *Computational Geometry in C*, Cambridge University Press, New York, 1994.

[20] Pollack, R., M. Sharir, and G. Rote, Computing the Geodesic Center of a Simple Polygon. *Discrete and Computational Geometry*, **4**, pp. 611-626, 1989.

[21] Preparata, F.P. and M.I. Shamos, *Computational Geometry – An Introduction.* Springer-Verlag, New York, 1985.

[22] Pribble, W.I., Molds for Reaction Injection, Structural Foam and Expandable Styrene Molding, in *Plastics Mold Engineering Handbook, Fourth Edition*, J. H. DuBois and W. I. Pribble (eds.), Van Nostrand Reinhold Company Inc., New York, 1987.

[23] Sarnak, N. and R.E. Tarjan, Planar Point Location using Persistent Search Trees, *Comm. of ACM*, **29**(7), pp. 669-679, 1986.

[24] Suri, S., On Some Link Distance Problem in a Simple Polygon. *IEEE Trans. on Robotics and Auto*, **6**(1), pp. 108-113, 1990.

[25] Suri, S., Computing Geodesic Furthest Neighbors in Simple Polygons. *J. of Comp. and Sys. Sciences*, **39**, pp. 220-235, 1989.

[26] Trochu, F., personal communication, November 1993.