Computing the External Geodesic Diameter of a Simple Polygon

by

David Samuel

School of Computer Science
McGill University, Montréal.

September 1988

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Master of Science.

Abstract

Given a simple polygon P of n vertices, we present an algorithm that finds the pair of points on the boundary of P that maximize the *external* shortest path between them. This path is defined as the *external* geodesic diameter of P. The algorithm takes $O(n^2)$ time and requires $O(n)$ space. Surprisingly, this problem is quite different from that of computing the *internal* geodesic diameter of P. While the *internal* diameter is determined by a pair of vertices of P, this is not the case for the *external* diameter. Finally, we show how this algorithm can be extended to solve the *all external geodesic furthest neighbours* problem.

Resumé

Ce travail présente un algorithme qui trouve une paire de points sur la frontière d'un polygone simple P de n coins, qui maximise le *trajectoire externe le plus court* entre les deux points. Ce trajectoire est définit par le *diamètre géodésique externe* de P. L'algorithme prend un temps $O(n^2)$ et requiert un champ $O(n)$. Fait surprennant, ce problème n'est pas semblable au problème de calculer le *diamètre géodésique interne*. Bien que le diamètre interne est determiné par une paire de coins de P, ce n'est pas le cas pour le diamètre externe. En fin, ce travail demontre comment cet algorithme peut être étendu pour résoudre le problème de *tous les voisins à la plus grande distance géodésique externe*.

Acknowledgements

I would like to thank my thesis advisor Professor Godfried Toussaint, for his invaluable assistance in my research. I would also like thank Mike Parker and Tom Shermer for the many happily distracting hours of discussion, and I would especially like to thank Michael Houle for being the great driving force in getting me to complete this thesis.

"What has it got in its pocketses?"

— Gollum —

Contents

Chapter 1

Introduction

Let P be a polygon described by a list of *vertices* $(v_0, v_1, \ldots, v_n)$, where the vertices v_0 and v_n are considered to be identical. The *edge* $e_i = (v_{i-1}, v_i)$ of P is the closed line segment connecting v_{i-1} and v_i, for $1 \leq i \leq n$. In addition, we say that P is *simple* if no two non-consecutive edges intersect, and consecutive edges intersect at a vertex. The collection of edges of P form the boundary $bd(P)$. The interior and the exterior of P shall be denoted as $int(P)$ and $ext(P)$, respectively. The polygon is considered to consist of the union of its boundary and its interior.

Let $\delta(p,q)$ be the Euclidean distance between points p and q. Given a polygon P, an *external geodesic path* between p_0 and p_k, denoted $\pi_e(p_0,p_k)$, is a polygonal chain with vertices $(p_0, p_1, \ldots, p_k)$ which both avoids $int(P)$ and minimizes the path length $\sum_{i=1}^{k}\delta(p_{i-1},p_i)$ (see Figure 1.1). This minimum path length shall be called the *external geodesic distance* between p_0 and p_k, and shall be denoted $\delta_e(p_0,p_k)$. Analogously, we define the exterior-avoiding *internal geodesic path* $\pi_i(p_0,p_k)$, and its associated *internal geodesic distance* $\delta_i(p_0,p_k)$.

Given a polygon P, an *external geodesic diameter* $\Delta_e(P)$ is an external geodesic path with endpoints p and q in P, such that

$$\delta_e(p,q) \geq \delta_e(p',q')\, \forall p',q' \in P.$$

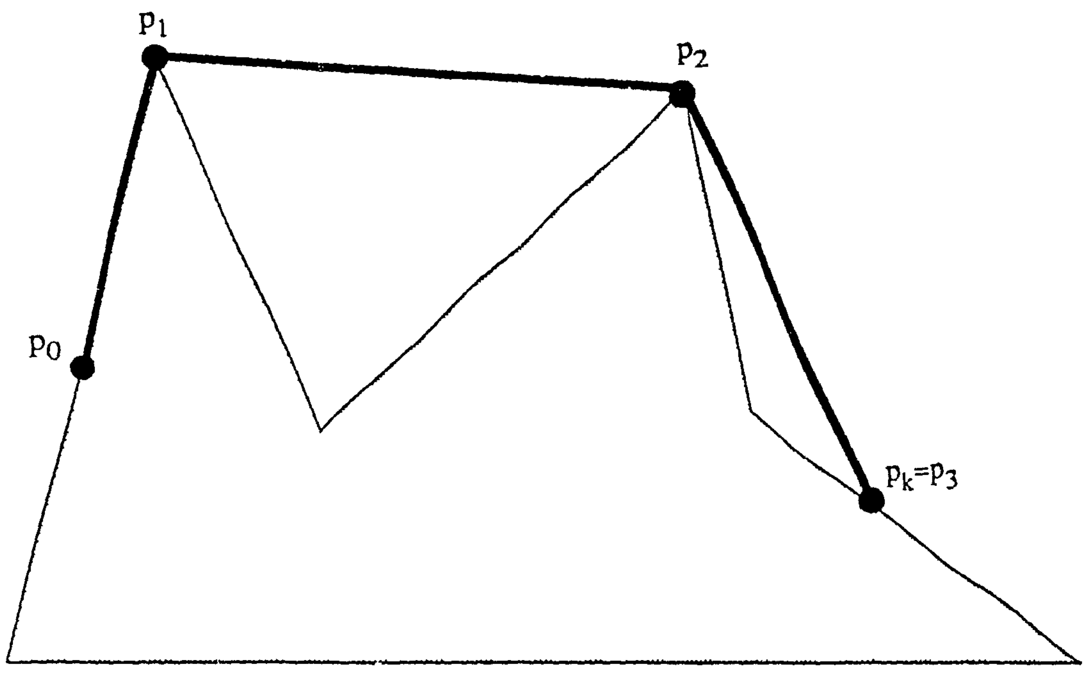


Figure 1.1: An external geodesic path

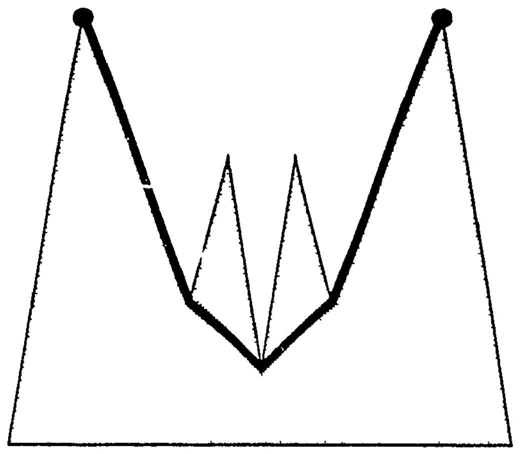
The length of this diameter is denoted by $\delta_e(P)$. Similarly, for the internal case, we define $\Delta_\iota(P)$ and $\delta_\iota(P)$ (see Figure 1.2).

Although Shamos [16] first proposed the problem of finding shortest paths, or geodesics, in a simple polygon, the first non-trivial results were due to Chazelle [3], who developed divide-and-conquer algorithms for many such problems. He used preprocessing of the polygons to solve the *internal geodesic distance*, *all internal geodesic path*, and *internal geodesic diameter* problems (which he called the *internal path*, *all internal path*, and *internal length* problems), in time $O(n)$ plus triangulation, $O(n^2)$, and $O(n^2)$, respectively. The *all internal geodesic path* problem is that of preprocessing a polygon such that the geodesic path between two query points can be determined in optimal time.

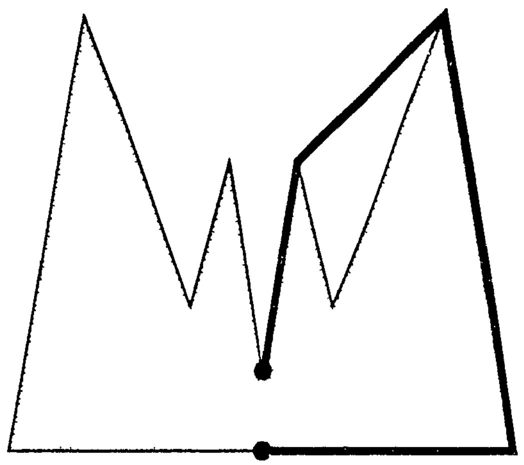
Suri [18] has subsequently improved upon one of these results with an $O(n \log n)$ algorithm for computing the internal geodesic diameter of a simple polygon.

Guibas, Hershberger *et al.* [6] give an algorithm for computing the *shortest path tree* of a polygon P with respect to a point $x \in int(P)$. This algorithm requires only linear time, assuming that a triangulation of P is supplied. The shortest path tree of a polygon P with respect to x is simply the union of all the geodesics from x to all vertices v of P. The shortest path tree can be used to answer queries about the length of the shortest path from x to the query point y in $O(\log n)$ time; the path itself can be computed in $O(\log n + k)$ time, where k is the number of links in the path. Guibas and Hershberger [5] also give a method of solving the *all internal geodesic path* problem in $O(n \log n)$ time.

Asano and Toussaint [2] have given an $O(n^3 \log n)$ algorithm which solves a related problem, that of finding the *geodesic center* of a simple polygon. The geodesic center is the point which minimizes the maximum geodesic distance to any point of P. Pollack, Rote and Sharir [12] have subsequently improved this with an algorithm that runs in



a)



b)

Figure 1.2: Internal and External Geodesic Diameters

$O(n \log n)$ worst-case time.

This thesis is divided into six chapters. The following chapter presents some definitions and lemmas about the behavior of external geodesics on polygons. Chapter three presents the algorithm of Guibas, Hershberger *et al.* [6] for computing the shortest path tree of a polygon P rooted at s. Chapter four presents Suri's [18] method for determining the internal geodesic diameter of a simple polygon. Chapter five presents the concept of a depth profile and gives our algorithms for determining the external geodesic diameter of a simple polygon and for solving the *all external geodesic furthest neighbours* problem. Finally, Chapter six contains some concluding remarks.

Chapter 2

Definitions and Preliminary Results

In this chapter we present some needed definitions and make a few observations about the behaviour of external geodesics on simple polygons.

A *convex polygon* is a polygon such that, for any two points in the polygon, the closed line segment joining the points is entirely contained in the polygon. The *convex hull* $CH(P)$ of a polygon P is defined as the minimum area convex polygon enclosing P. We define a *pocket* Q of P as a minimal polygonal chain of edges and vertices of P, whose edges are not coincident with convex hull edges, and whose endpoints are the only vertices of Q lying on the boundary of $CH(P)$ (see Figure 2.1). We define the *lid* of a pocket as the line segment (convex hull edge) that connects the endpoints of the pocket. A pocket together with its lid forms a *pocket polygon.*

Let x and y be distinct points on the boundary of polygon P, such that x and y are not contained in a common pocket of P. We define a *circuit* of P with respect to x and y as the shortest closed polygonal chain containing x and y that entirely encloses $int(P)$ (see Figure 2.2). More specifically and without loss of generality, we can assume that our circuit is $(p_0, p_1, \ldots, p_k)$, where $p_0 = p_k = x$ and $p_i = y$. The points x and y can then be said to delimit two *semi-circuits* $(p_0 = x, p_1, \ldots, p_i =$

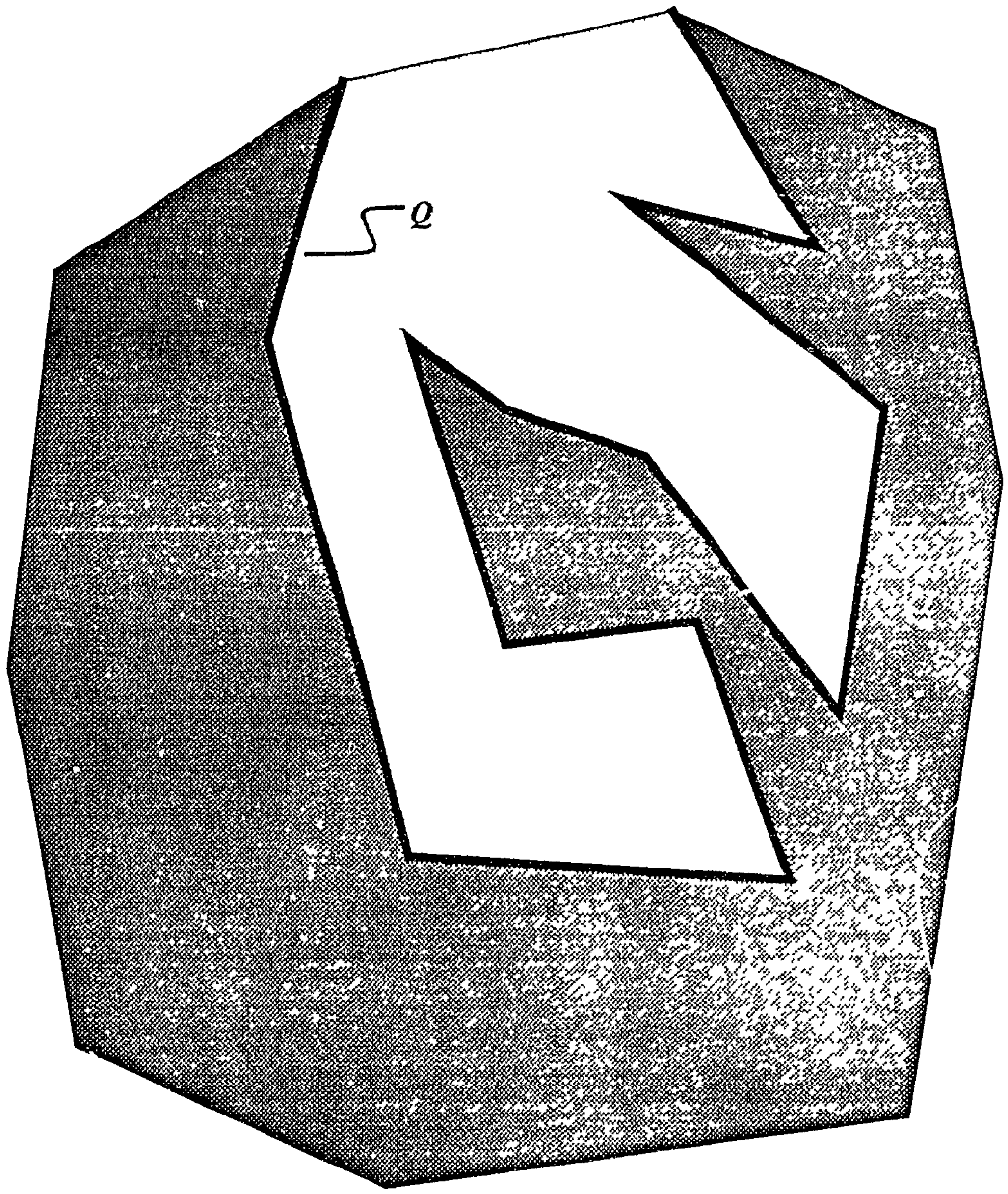


Figure 2.1: A pocket and its pocket polygon

y) and ($p_i = y, p_{i+1}, \ldots, p_k = x$). It should be noted that $\pi_e(x, y)$ is one of these semi-circuits.

A *furthest neighbour* $\phi(v)$ of a point v contained in a set of points Q, is defined as

$$\phi(v) \in Q \Longrightarrow \delta(v, \phi(v)) = \max_{x \in Q} \delta(v, x).$$

A *geodesic furthest neighbour* $\phi_\iota(v)$ of a vertex v in a polygon P, is defined as

$$\phi_\iota(v) \in P \Longrightarrow \delta_\iota(v, \phi(v)) = \max_{x \in P} \delta_\iota(v, x).$$

The *all geodesic furthest neighbours* problem is simply that of determining the geodesic furthest neighbour for each of the vertices in a polygon. An *external geodesic furthest neighbour* $\phi_e(v)$ of a vertex v on the boundary, $bd(P)$, of a simple polygon P is

$$\phi_e(v) \in bd(P) \Longrightarrow \delta_e(v, \phi(v)) = \max_{x \in bd(P)} \delta_e(v, x).$$

The *all external geodesic furthest neighbours* problem is the problem of determining the external geodesic furthest neighbour of each of the vertices of the polygon.

To prove the first of our lemmas, we observe that a polygon with no pockets (*i.e.* a convex polygon) has an infinite number of pairs of points that realize the external geodesic diameter, and that this diameter is merely one-half of the length of its perimeter (see Figure 2.3).

Lemma 2.1 *If a simple polygon P has only one pocket Q, then at least one point of a pair that realizes some $\Delta_e(P)$ must lie on Q.*

Proof Assume the contrary. Then the endpoints of all external geodesic diameters must lie on the boundary of $CH(P)$. Therefore, $\delta_e(P) = \delta_e(CH(P))$. Let m be the midpoint of the pocket lid. Then there exists a diameter $\Delta_e(CH(P))$, which has m as one endpoint and $m' \in P$ as the other. Let q_0 be a point of Q on the perpendicular bisector of the lid. Let a and b be the endpoints of the lid (see Figure 2.4). Then the

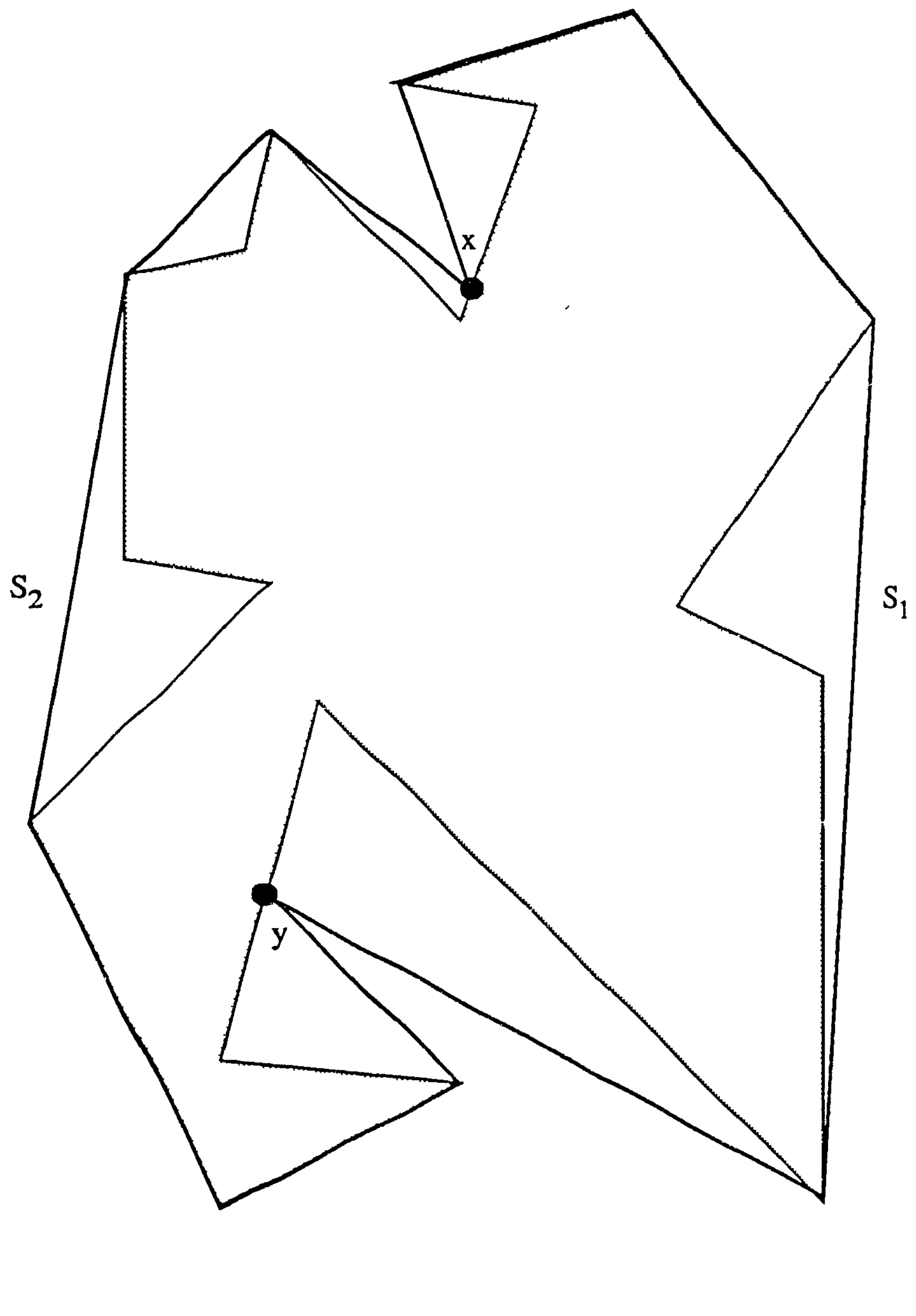


Figure 2.2: Circuits and semi-circuits with respect to x and y

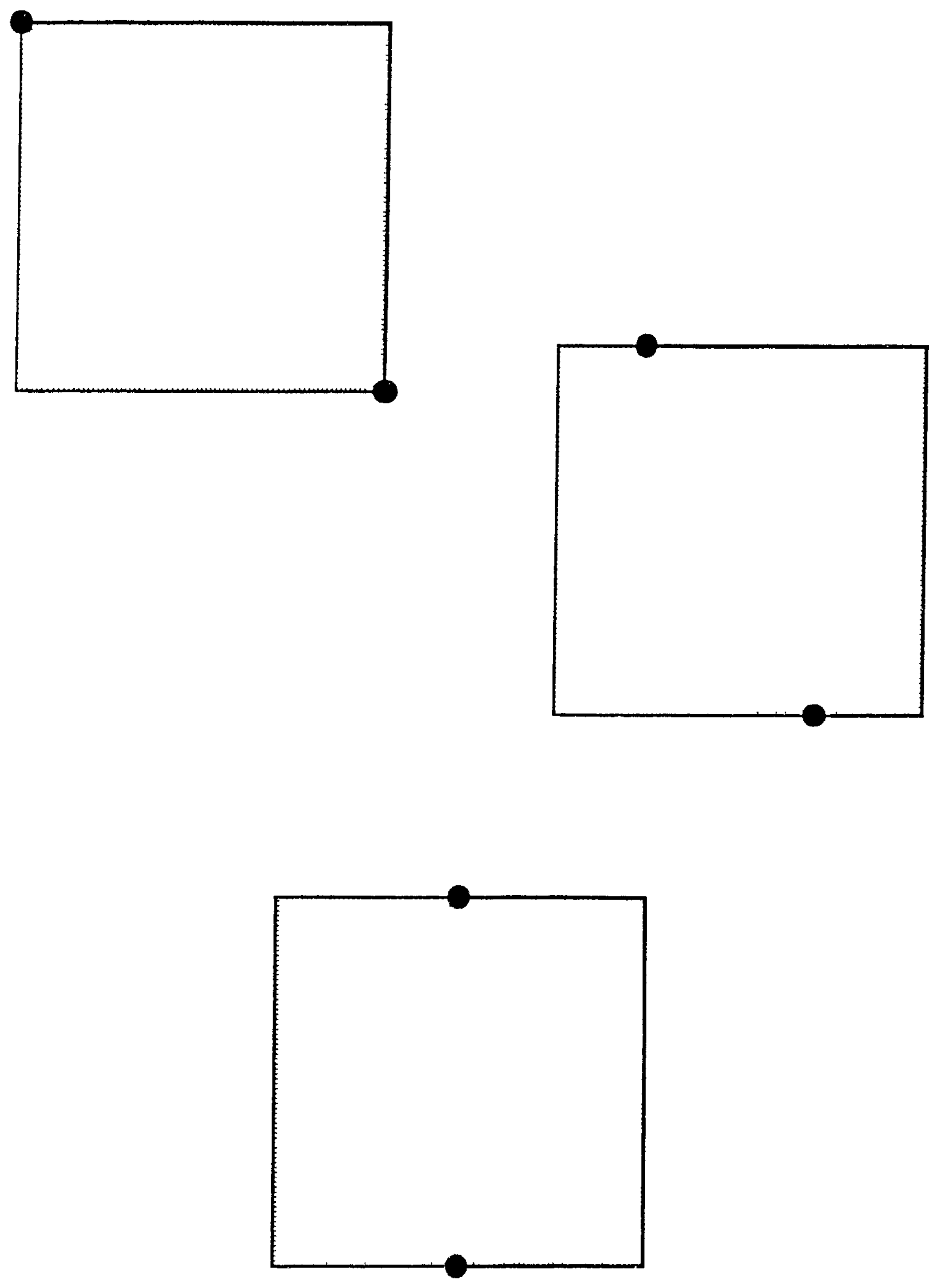


Figure 2.3: External geodesic diameters of convex polygons

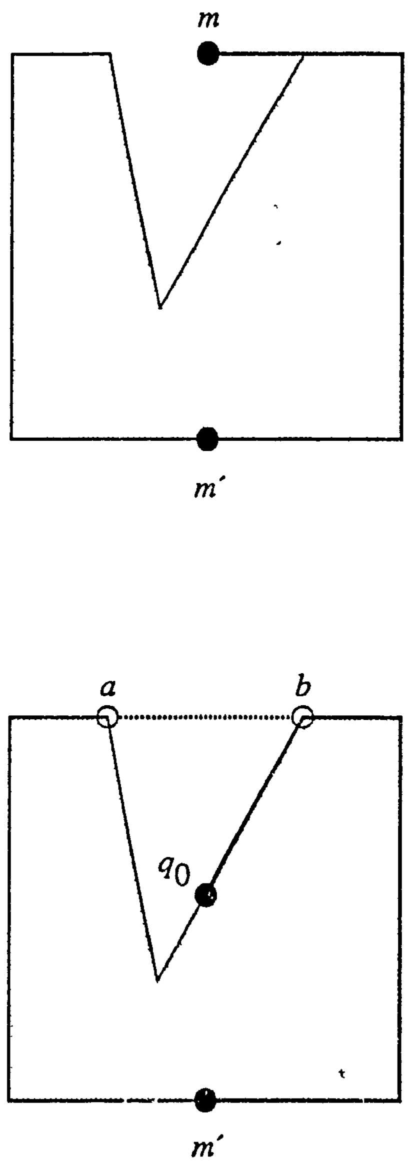


Figure 2.4: Illustration for the proof of Lemma 2.1

following holds:

$$\delta_\varepsilon(a, q_0) > \frac{1}{2}\delta(a, b) \text{ and } \delta_\varepsilon(b, q_0) > \frac{1}{2}\delta(a, b).$$

Then clearly $\delta_\varepsilon(q_0, m') > \delta_\varepsilon(m, m') = \delta_\varepsilon(P)$, which is a contradiction. □

It is in fact possible that the endpoints of an external geodesic diameter lie within a common pocket, as illustrated in Figure 2.5.

Lemma 2.2 *Let (q_0, q_0') be a pair of points of polygon P that realizes a diameter $\Delta_\varepsilon(P)$, such that q_0 is in pocket Q and q_0' is contained in no pocket. Then the two semi-circuits of P defined by q_0 and q_0' are of equal length.*

Proof Assume the contrary Let S_1 and S_2 be the two semi-circuits defined by q_0 and q_0', with lengths l_1 and l_2, respectively. Without loss of generality, we can assume that $l_2 > l_1 = \delta_\varepsilon(P)$. For all $\epsilon > 0$, there exists some point $x \in S_2$ such that $0 < \delta_\varepsilon(q_0', x) < \epsilon$. Then for ϵ sufficiently small, $\delta_\varepsilon(x, q_0) = \delta_\varepsilon(P) + \epsilon$, which is a contradiction. □

Lemma 2.3 *Let (q_0, q_0') be a pair of points of polygon P that realizes a diameter $\Delta_\varepsilon(P)$, such that q_0 is in pocket Q and q_0' is contained in no pocket. Then q_0 will be a point which attains*

$$\max_{q \in Q} (\delta_\varepsilon(q, a) + \delta_\varepsilon(q, b)),$$

where a and b are the endpoints of the lid of Q.

Proof Assume the contrary. Of the two semi-circuits defined by a and b, let S be that semi-circuit not containing the lid of Q. Let l be the length of S. If we let q_0' be the other endpoint of $\Delta_\varepsilon(P)$, then

$$\delta_\varepsilon(P) = \min\{\delta_\varepsilon(q_0, a) + \delta_\varepsilon(q_0', a), \delta_\varepsilon(q_0, b) + \delta_\varepsilon(q_0', b)\}.$$

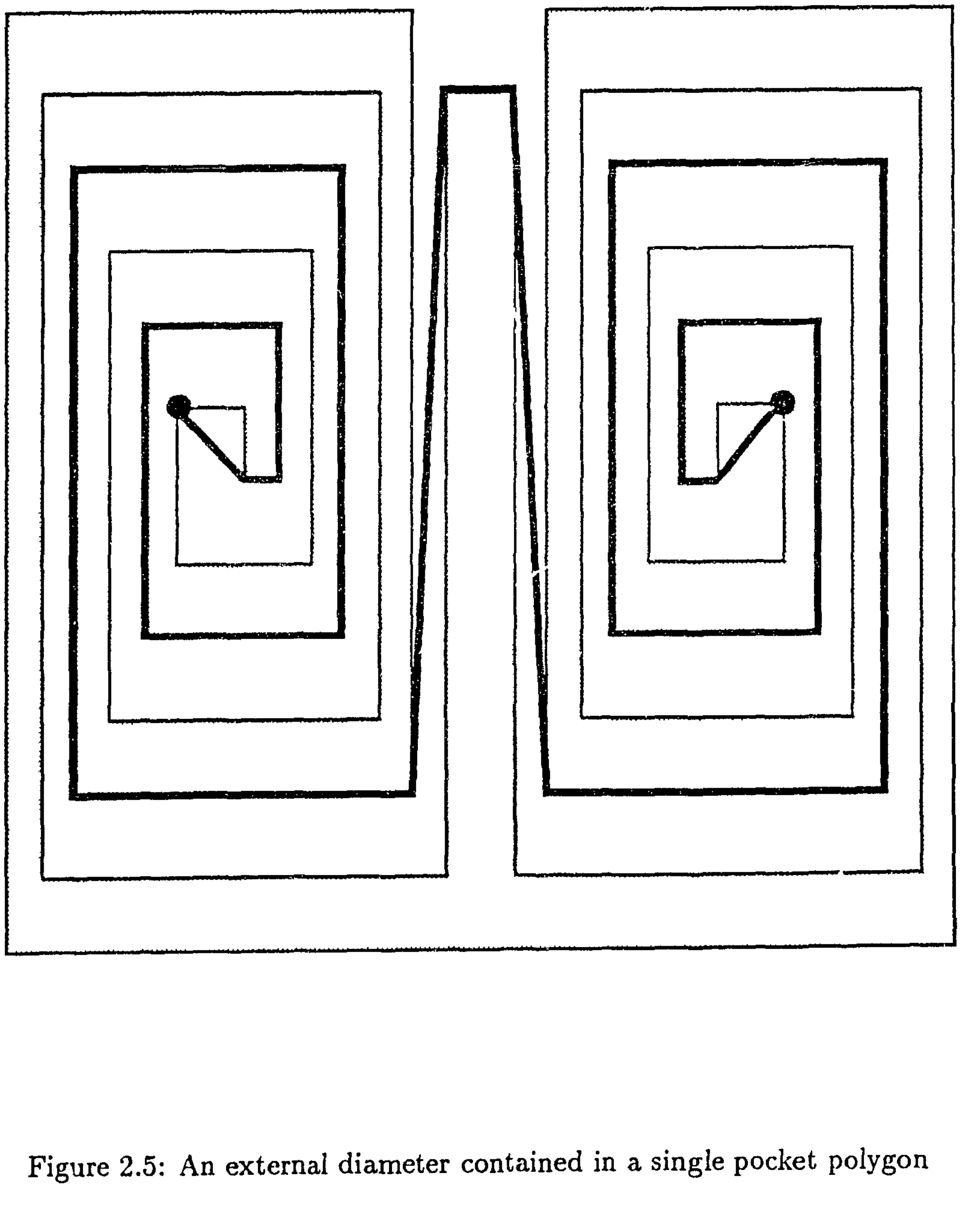


Figure 2.5: An external diameter contained in a single pocket polygon

By Lemma 2.2, the two quantities are equal, and thus also equal to their average:

$$\delta_\varepsilon(P) = \frac{1}{2}(\delta_\varepsilon(q_0, a) + \delta_\varepsilon(q_0', a) + \delta_\varepsilon(q_0, b) + \delta_\varepsilon(q_0', b)).$$

Since [illegible] 's contained in S, then

$$\delta_\varepsilon(P) = \frac{1}{2}(\delta_\varepsilon(q_0, a) + \delta_\varepsilon(q_0, b) + l),$$

Consider $x \in Q$ such that $\delta_\varepsilon(x, a) + \delta_\varepsilon(x, b) > \delta_\varepsilon(q_0, a) + \delta_\varepsilon(q_0, b)$. Let $x' \in CH(P)$ such that $\delta_\varepsilon(x, x')$ is maximized. Similar to the methods above, we obtain

$$\delta_\varepsilon(x, x') = \frac{1}{2}(\delta_\varepsilon(x, a) + \delta_\varepsilon(x, b) + l),$$

which implies that $\delta_\varepsilon(x, x') > \delta_\varepsilon(P)$. Clearly, this leads to a contradiction. □

Lemma 2.4 *Given a pocket Q with endpoints a and b, the point q_0 that realizes*

$$\max_{q \in Q} (\delta_\varepsilon(q, a) + \delta_\varepsilon(q, b))$$

is a vertex of Q.

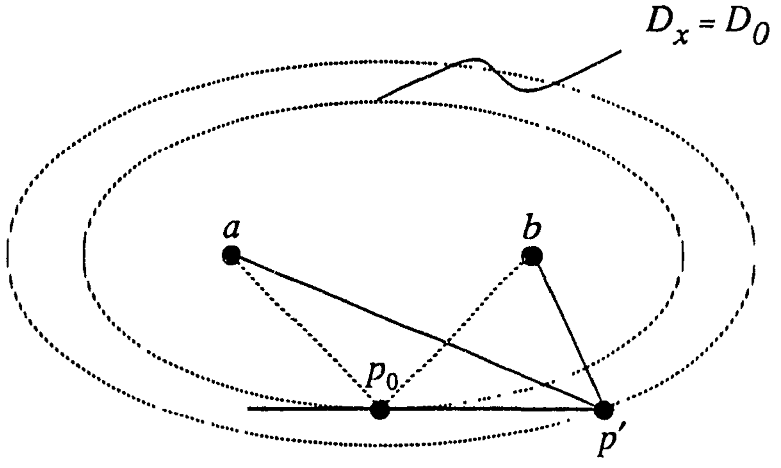
Proof Assume the contrary. Let $D_0 = \delta_\varepsilon(q_0, a') + \delta_\varepsilon(q_0, b')$, and let $D_x = \delta_\varepsilon(x, a') + \delta_\varepsilon(x, b')$ for some $x \in Q$, $x \neq q_0$, where a' and b' are the last vertices on the paths from a and b, respectively. Note that a' and b' may be the same vertex.

The locus of points x such that $D_x = D_0$ defines an ellipse. Let l be the unique edge of Q containing q_0. This gives us two cases:

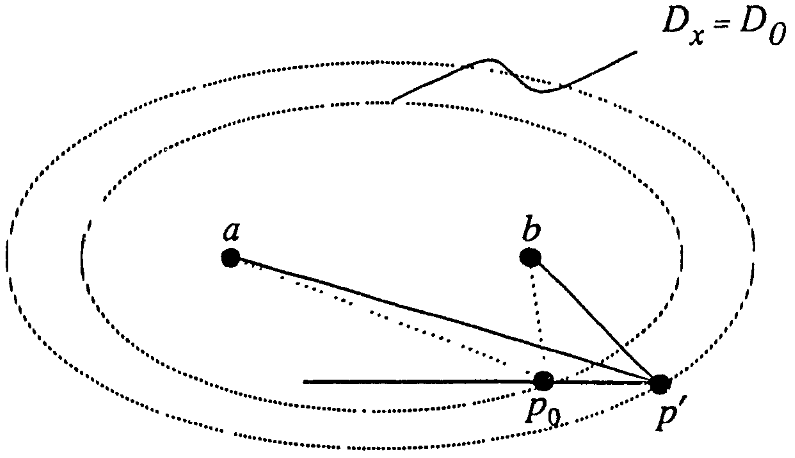
Case 1: l is tangent to the ellipse defined by $D_x = D_0$ (see Figure 2.6(a)). For all $\epsilon > 0$, there exists some $x \in l$ such that $0 < \delta_\varepsilon(q_0, x) < \epsilon$. Then for ϵ sufficiently small, we have

$$D_x = \delta_\varepsilon(x, a') + \delta_\varepsilon(x, b') > D_0,$$

which is a contradiction.



(a)



(b)

Figure 2.6: Illustration for Lemma 2.4

Case 2: l intersects the ellipse at q_0 (see Figure 2.6(b)). This case is similar to Case 1, except we restrict x to lie on the side of q_0 opposite that of the perpendicular bisector of a' and b'. Again we arrive at a contradiction. □

The preceding lemmas may be used to indicate how to handle some of the situations in which the diameter of a polygon P may be found. We have four types of candidates for our diameter:

1. neither endpoint is in a pocket,
2. both endpoints are in the same pocket,
3. one endpoint is in a pocket and the other is on the convex hull of P, and
4. the two endpoints are in different pockets.

The first case can only occur if the polygon has no pockets, as was shown in Lemma 2.1. Candidates for type 2 diameters can be generated by finding the internal geodesic diameters of all the pocket polygons of P. For type 3 diameters, it suffices to check the vertices that attain $\max_{q \in Q} (\delta_\varepsilon(q,a) + \delta_\varepsilon(q,b))$ for each pocket Q, with endpoints a and b, as was shown in Lemmas 2.3 and 2.4.

It is the last type that requires the most involved analysis. As we have seen, it is possible that one endpoint of the diameter does not lie on a vertex of the polygon. However, even though it is possible that an external geodesic diameter can be realized with neither endpoint on a vertex, such a diameter will not be unique, and there will exist an equivalent diameter that has at least one endpoint on a vertex.

Lemma 2.5 *Given some $\Delta_\varepsilon(P)$ with neither of its endpoints q_0 and q_0' on a vertex, then the two semi-circuits S_1 and S_2 defined by q_0 and q_0' must be of equal length.*

Proof The case where the polygon has no pockets has already been considered. Otherwise, if there exists at least one pocket, by Lemma 2.1 at least one endpoint of $\Delta_\varepsilon(P)$ (say q_0) is in some pocket Q. Let l_1 and l_2 be the lengths of S_1 and S_2 respectively. Assume that $l_1 \neq l_2$.

Without loss of generality, assume $l_1 < l_2$. Let a' and b' be the vertices adjacent to q_0 in S_2 and S_1, respectively. Let ξ be the edge of Q containing q_0. Consider θ, the angle between ξ and the line segment $\overline{aq}_0$, measured clockwise from ξ. We can have three cases:

1. $\theta < \frac{\pi}{2}$
2. $\theta = \frac{\pi}{2}$
3. $\theta > \frac{\pi}{2}$

Case 1: If $\theta < \frac{\pi}{2}$, then we can connect a' to p, a point on ξ which is chosen to be distance ϵ from q_0, and outside the angle θ, for some $\epsilon > 0$ which is sufficiently small (see Figure 2.7(a)).

$$\begin{aligned}
(\delta_\varepsilon(a',p))^2 &= (x+\epsilon)^2 + y^2 \\
&= x^2 + y^2 + 2x\epsilon + \epsilon^2 \\
&> (\delta_\varepsilon(a',q_0))^2 \\
\delta_\varepsilon(a',p) &> \delta_\varepsilon(a',q_0)
\end{aligned}$$

which is a contradiction. It should be noted that a situation can arise whereby there is a vertex a'' of S_1 which is collinear with $\overline{aq}_0$. This situation can be resolved as above by choosing a'' in place of a'.

Case 2: Similar to Case 1 except that p can lie to either side of q_0 (see Figure 2.7(b)).

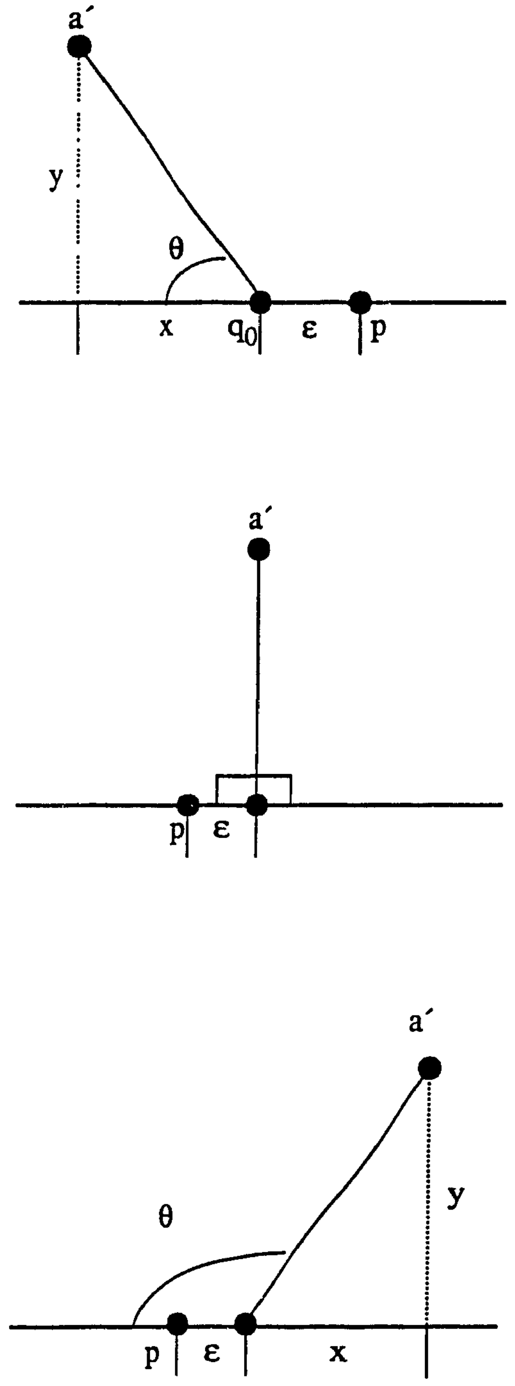


Figure 2.7: The three cases of θ

Case 3: Similar to Case 1, except that p has to lie inside the angle θ (see Figure 2.7(c)).

These three contradictions imply that $l_1 = l_2$. □

Lemma 2.6 *If $\exists\,\Delta_\varepsilon(P)$ realized by points q_0 and q_0', where neither q_0 nor q_0' is a vertex, then $\exists\, p_0$ and p_0' such that $\delta_\varepsilon(p_0, p_0') \geq \delta_\varepsilon(q_0, q_0')$, and at least one of p_0 and p_0' is a vertex.*

Proof If q_0' is not contained in a pocket of P, the result follows from Lemmas 2.3 and 2.4. Otherwise, let Q and Q' be the pockets containing q_0 and q_0' respectively. Let $\xi \in Q$ and $\xi' \in Q'$ be the edges containing q_0 and q_0' respectively. Consider the semi-circuits $S_1 = (q_0, \beta, \ldots, \beta', q_0')$ and $S_2 = (q_0', \alpha', \ldots \alpha, q_0)$.

We note that α and β cannot lie on the same side of the perpendicular to ξ through q_0, since in that case we can simply choose a point on ξ which is ε to the other side and extend our path, leading to a contradiction. Similarly for α' and β'. We also note that neither α nor β can lie on the perpendicular through q_0 by an analogous argument. Similarly for α' and β'.

We define a to be the perpendicular distance from α to the line through ξ, and label the point at which the perpendicular meets ξ (or its extension), as O. We define b to be to be the distance from O to q_0 and parameterize it with x. We parameterize the perpendicular distance by y. Similarly, we get d to be the perpendicular distance from β to ξ (or its extension), and c to be the distance from said perpendicular to q_0. Applying similar treatment to pocket Q' yields point O', distances e, f, g, and h, and parameters u and v (see Figure 2.8).

Without loss of generality, we assume that $h > e$. We move one end-point of $\Delta_\varepsilon(P)$ some small amount Δx and observe the behavior of $r_1(x)$ and $r_2(x)$.

$$r_1(x) = \sqrt{a^2+b^2} + \sqrt{e^2+f^2} - \sqrt{a^2+x^2}$$

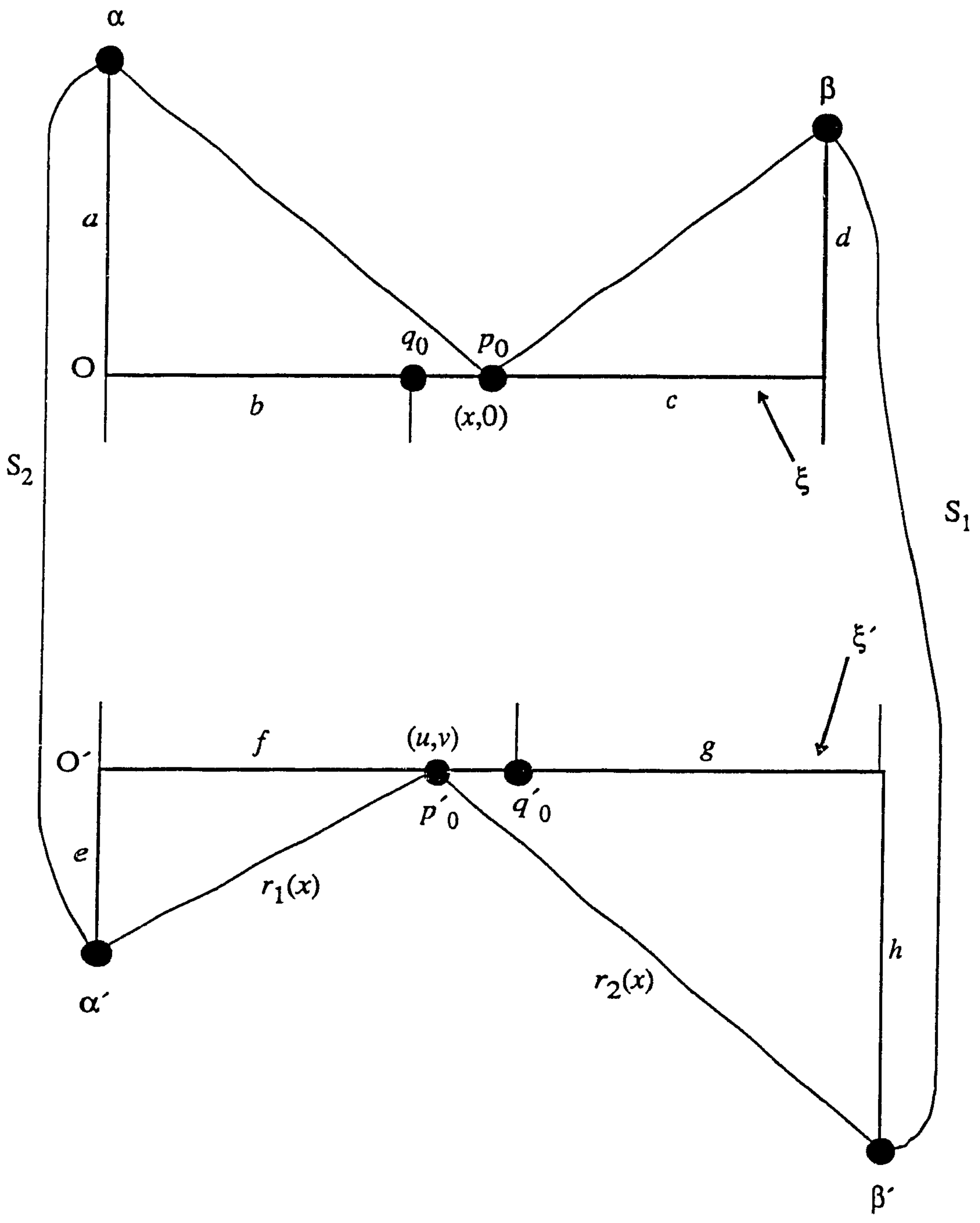


Figure 2.8: Illustration for the proof of Lemma 2.6

$$r_2(x) = \sqrt{c^2+d^2} + \sqrt{g^2+h^2} - \sqrt{d^2+(b+c-x)^2}.$$

From the Pythagorean theorem, we have

$$r_1(x)^2 = u^2 + (v-e)^2 \tag{2.1}$$

$$r_2(x)^2 = (u-(f+g))^2 + (v-h)^2. \tag{2.2}$$

From (2.1), we get $u = \pm\sqrt{r_1^2 - (v-e)^2}$; substituting this into (2.2), we obtain

$$\left(\pm\sqrt{r_1^2-(v-e)^2} - (f+g)\right)^2 + (v-h)^2 = r_2^2$$

$$(r_1^2-(v-e)^2) \pm 2(f+g)\sqrt{r_1^2-(v-e)^2} + (f+g)^2 + (v-h)^2 = r_2^2.$$

Rearranging gives

$$\begin{aligned}\pm\sqrt{r_1^2-(v-e)^2} &= \frac{r_2^2 - r_1^2 + (v-e)^2 - (f+g)^2 - (v-h)^2}{2(f+g)}\\ &= \frac{r_2^2-r_1^2-(f+g)^2+e^2-h^2}{2(f+g)} + v\left(\frac{h-e}{f+g}\right),\end{aligned}$$

since we can assume that $f+g>0$, because otherwise the point is already at a vertex. If we define constants C and Q as

$$\begin{aligned}C &= \frac{(h-e)}{(f+g)}\\ Q &= \frac{(r_2^2-r_1^2-(f+g)^2+e^2-h^2)}{2(f+g)}\end{aligned}$$

we have $\pm\sqrt{r_1^2-(v-e)^2} = vC+Q$. Solving for v, we have

$$\begin{aligned}r_1^2-(v-e)^2 &= v^2C^2+2vCQ+Q^2\\ r_1^2-(v^2-2ve+e^2) &= v^2C^2+2vCQ+Q^2\end{aligned}$$

$$v^2(C^2+1) + 2v(CQ-e) + (Q^2+e^2-r_1^2) = 0$$

which in turn leads to

$$\begin{aligned} v &= \frac{-2(CQ-e) \pm \sqrt{4(CQ-e)^2 - 4(C^2+1)(Q^2+e^2-r_1^2)}}{2(C^2+1)} \\ &= \frac{-(CQ-e) \pm \sqrt{(CQ-e)^2 - (C^2+1)(Q^2+e^2-r_1^2)}}{(C^2+1)}. \end{aligned}$$

Because of the construction, v must have at least one real value, which means that

$$\sqrt{(CQ-e)^2 - (C^2+1)(Q^2+e^2-r_1^2)} \in \Re.$$

We want the smaller value of v, which is

$$v = \frac{-(CQ-e) - \sqrt{(CQ-e)^2 - (C^2+1)(Q^2+e^2-r_1^2)}}{(C^2+1)}.$$

Since we want $v \geq 0$, we must have

$$-(CQ-e) = (e-CQ) > 0$$

and

$$Q^2 + e^2 - r_1^2 \geq 0.$$

If these two conditions are satisfied, the existence of a non-negative value for v implies that we can move by some distance Δx so as not to decrease the path length.

To show that $e - CQ > 0$, we consider that

$$\begin{aligned} e - CQ &= e - \left[\frac{h-e}{f+g}\right]\left[\frac{e^2 - h^2 + r_2^2 - r_1^2 - (f+g)^2}{2(f+g)}\right] \\ &= \frac{2e(f+g)^2}{2(f+g)^2} - \frac{1}{2(f+g)^2}\left[he^2 - h^3 + hr_2^2 - hr_1^2 - h(f+g)^2\right. \\ &\qquad \left. - e^3 + eh^2 - er_2^2 + er_1^2 + e(f+g)^2\right] \\ &= \frac{1}{2(f+g)^2}\left[-he^2 + h^3 - hr_2^2 + hr_1^2 + h(f+g)^2\right. \\ &\qquad \left. + e^3 - eh^2 + er_2^2 - er_1^2 + e(f+g)^2\right]. \end{aligned}$$

Since $f+g>0$ as above, we can discard the $1/(2(f+g)^2)$ term. The remaining term evolves under our tender ministrations as follows:

$$\begin{aligned}2(f+g)^2(e-CQ) &= (h^2-e^2)(h-e)+(h-e)(r_1^2-r_2^2)+(h+e)(f+g)^2 \\ &= (h-e)\left[(r_1^2-e^2)-(r_2^2-h^2)+(f+g)^2\right]+2e(f+g)^2.\end{aligned}$$

If we now let

$$\begin{aligned}f_*^2 &= r_1^2-e^2 \\ g_*^2 &= r_2^2-h^2\end{aligned}$$

this becomes

$$2(f+g)^2(e-CQ)=(h-e)\left[f_*^2-g_*^2+(f+g)^2\right]+2e(f+g)^2.$$

If we now notice that $f_*^2 \to f^2$ and $g_*^2 \to g^2$ as $\Delta x \to 0$, and recall that we assumed that $h>e$, we obtain

$$\begin{aligned}2(f+g)^2(e-CQ) &= (h-e)\left[f^2-g^2+(f+g)^2\right]+2e(f+g)^2 \\ &= (h-e)\left[2f^2+2fg\right]+2e(f+g)^2 \\ &> 0.\end{aligned}$$

To show that $Q^2+e^2-r_1^2 \geq 0$, we consider that

$$Q^2+e^2-r_1^2=(Q+\sqrt{r_1^2-e^2})(Q-\sqrt{r_1^2-e^2}).$$

If we let K_+ be the first and K_- be the second terms on the right hand side, we note that the desired relationship holds if K_+ and K_- have the same sign. This is obviously true when $K_+ \leq 0$ or $K_- \geq 0$:

$$K_+=\left[\frac{e^2-h^2+r_2^2-r_1^2-(f+g)^2}{2(f+g)}\right]+\sqrt{r_1^2-e^2}.$$

To simplify this, we let $K_+^* = 2(f+g)K_+$. Then we have

$$K_+^* = e^2 - h^2 + r_2^2 - r_1^2 - (f+g)^2 + 2(f+g)\sqrt{r_1^2 - e^2}$$

We once again let

$$\begin{aligned} f_*^2 &= r_1^2 - e^2 \\ g_*^2 &= r_2^2 - h^2 \end{aligned}$$

resulting in

$$\begin{aligned} K_+^* &= g_*^2 - f_*^2 - (f+g)^2 + 2(f+g)f_* \\ &= g_*^2 - f_*^2 - f^2 - 2fg - g^2 + 2ff_* + 2gf_* \\ &= (g_*^2 - g^2) - (f - f_*)^2 - 2g(f - f_*) \\ &= (g_* - g)(g_* + g) + 2g(f_* - f) - (f_* - f)(f_* - f) \\ \frac{K_+^*}{\Delta x} &= \frac{g_* - g}{\Delta x}(g_* + g) + 2g\frac{f_* - f}{\Delta x} - (f_* - f)\frac{f_* - f}{\Delta x}. \end{aligned}$$

Noticing that this matches the definition of the derivative, we let $\Delta x \longrightarrow 0$ and write

$$\lim_{\Delta x \to 0} \frac{K_+^*}{\Delta x} = 2g[g_*' + f_*'].$$

Since $g_* = \sqrt{r_2^2 - h^2}$,

$$g_*' = \frac{r_2}{\sqrt{r_2^2 - h^2}} r_2' = \frac{r_2}{g_*} r_2' = \frac{r_2}{g} r_2'$$

when Δx is sufficiently small. Similarly, $f_*' = (r_1/f)r_1'$. Since the derivatives of r_1 and r_2 are

$$\begin{aligned} r_1' &= -\left[\frac{x}{\sqrt{a^2 + x^2}}\right] \\ r_2' &= -\left[\frac{x - (b+c)}{\sqrt{d^2 + (b+c-x)^2}}\right] \end{aligned}$$

we can then write f'_* and g'_* as

$$\begin{aligned} f'_* &= -\frac{r_1}{f}\left[\frac{x}{\sqrt{a^2+x^2}}\right] \\ g'_* &= -\frac{r_2}{g}\left[\frac{x-(b+c)}{\sqrt{d^2+(b+c-x)^2}}\right]. \end{aligned}$$

Evaluating these at $x = b$ yields

$$\begin{aligned} f'_* &= \left[\frac{\sqrt{e^2+f^2}}{f}\right]\left[\frac{b}{\sqrt{a^2+b^2}}\right] \\ g'_* &= \left[\frac{\sqrt{g^2+h^2}}{g}\right]\left[\frac{c}{\sqrt{c^2+d^2}}\right] \end{aligned}$$

$$\lim_{\Delta x \to 0}\frac{K^*_+}{\Delta x} = 2g\left[\left(\frac{\sqrt{g^2+h^2}}{g}\right)\left(\frac{c}{\sqrt{c^2+d^2}}\right) - \left(\frac{\sqrt{e^2+f^2}}{f}\right)\left(\frac{b}{\sqrt{a^2+b^2}}\right)\right].$$

Therefore, we see that we can always choose a direction to move in so as to make K^*_+ negative, which in turn implies

$$Q^2 + e^2 - r_1^2 \geq 0.$$

Which implies that we can always move q_0 in some direction so as not to decrease the lengths of S_1 and S_2. In fact, we can move to the nearest vertex in said direction, since once we move a distance Δx, the path lengths are no longer equal, and Lemma 2.5 applies. □

We are now in a position to prove the main theorem of this chapter.

Theorem 2.7 *An external geodesic diameter of a simple polygon P can always be realized with at least one endpoint on a vertex of P.*

Proof If P has no pockets, then the theorem holds trivially. If P has at least one pocket, then the theorem follows from Lemma 2.6. □

Lemma 2.8 *Let Q be a pocket of polygon P, with endpoints a* and *b. Then*

$$\max_{q \in Q} |\delta_\varepsilon(a,q) - \delta_\varepsilon(b,q)| \leq \delta(a,b).$$

Proof Assume the contrary. That implies that there exists some $q_0 \in Q$ such that $|\delta_\varepsilon(a,q_0) - \delta_\varepsilon(b,q_0)| > \delta(a,b)$. Without loss of generality, assume that $\delta_\varepsilon(a,q_0) > \delta_\varepsilon(b,q_0)$. This implies that $\delta_\varepsilon(a,q_0) > \delta(a,b) + \delta_\varepsilon(b,q_0)$. But then we have an external polygonal chain from a to q_0 through b whose length is less than $\delta_\varepsilon(a,q_0)$, which is a contradiction. □

Chapter 3

Shortest Path Tree

If s is a vertex of a polygon P, then the *shortest path tree* of P with respect to s is simply $\bigcup_v \pi_t(s,v)$ for all vertices $v \in P$. This section deals with Guibas, Hershberger *et al.*'s [6] construction of the shortest path tree for a source vertex s of a polygon P. The algorithm for the construction requires linear plus triangulation time and is an extension of an algorithm due to Lee and Preparata [10].

For the sake of completeness, we now present an overview of the algorithm for the construction of the shortest path tree, the full details of which may be found in [6].

3.1 Observations

Let T be a triangulation of the interior of P. Then the planar dual D of T is a tree, whose vertices have degree at most 3. This implies that for all vertices x of P, there exists a unique minimal path π in D from a triangle containing s to another triangle containing x. This gives an ordered sequence of diagonals $d_1, \ldots, d_l$ corresponding to the path between s and x, each diagonal being the boundary between two triangles corresponding to adjacent points in π. Therefore, each d_i divides P into two parts, one containing s and the other containing x; therefore the geodesic path from s to x corresponding to the dual tree path π must intersect each d_i exactly once.

Then for $d = \overline{uw}$ being a diagonal or edge of P, we can construct a *funnel* $F_{\overline{uw}}$ (see Figure 3.1), which is the union of those parts of $\pi_t(s,u)$ and $\pi_t(s,w)$ which are not common to both. As in [6], the point common to both geodesic paths furthest from s we call the *cusp* (denoted by a in Figure 3.1), and note that in [10] it was proved that the portions from a to u and from a to w are outward-convex. Then if d is a diagonal of P in T, we note that exactly one triangle of T contains d but does not (otherwise) intersect the area bounded by d and the funnel $F_{\overline{uw}}$. Let v be the third vertex of this triangle, that is, the vertex which is neither u nor w. Then we note that the shortest path from s to v must start with the shortest path from s to a and then either continue along the straight-line segment $\overline{av}$ (if this segment does not intersect $F_{\overline{uv}}$) or follow one branch of the funnel until it reaches a vertex y at which the line segment $\overline{yv}$ is tangent to $F_{\overline{uw}}$. These observations are due to [10], and form the basis of their algorithm and its extension in [6].

3.2 Shortest Path Tree Algorithm

This algorithm first triangulates P. Then each diagonal $\overline{uw}$ is examined in turn, maintaining a *current* funnel $F = F_{\overline{uw}}$ as a list of vertices $(u_l, u_{l-1}, \ldots, u_1, a, w_1, \ldots, w_k)$ stored as a *finger tree* ([7] and [8]). This structure is essentially a search tree with auxiliary markers called *fingers*, which support searching for an element in $O(\log d)$ time, where d is the distance from x to the nearest finger, and also allows splitting the tree into two trees at an element x in amortized time $O(\log d)$. The cusp $CUSP(F)$ of the current funnel is also maintained.

The algorithm begins by initializing F to contain s and an adjacent vertex v_1, with $CUSP(F) = s$. It then proceeeds recursively as follows:

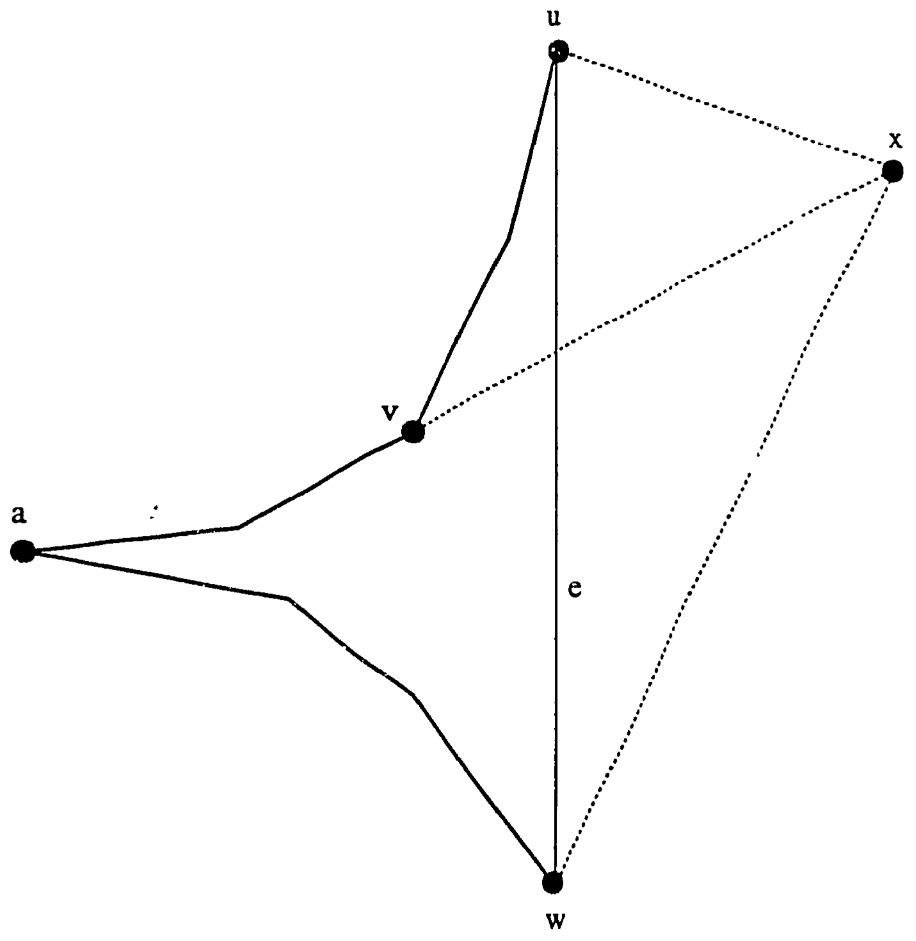


Figure 3.1: A funnel for diagonal $\overline{uw}$

Algorithm $PATH(F)$

Let u and w be the ends of F, with $a = CUSP(F)$. Let v be the third vertex of the triangle containing $\overline{uw}$ but not intersecting the area bounded by F and $\overline{uw}$ (as in the description above).

(1) Search F (with binary search) for a vertex y as in the description above, at which $\overline{yv}$ is tangent to F if such a vertex exists, or a, otherwise. Then split F at y and create two new funnels $F_1 = (u, \ldots, y, v)$ and $F_2 = (v, y, \ldots, w)$. Set $CUSP(F_1)$ to y, if y is on the path from a to u or to a otherwise; similarly, set $CUSP(F_2)$ to y, if y is on the path from a to w or to a otherwise.

(2) Concatenate $\overline{yv}$ with the path from s to y to obtain the path from s to v. (This is done by storing a pointer to y at v; the collection of such pointers gives the desired path tree.)

(3) If the segment $\overline{uv}$ is a diagonal of P (as opposed to an edge), call $PATH(F_1)$ recursively.

(4) Similarly, if the segment $\overline{wv}$ is a diagonal of P, call $PATH(F_2)$ recursively.

For the correctness and complexity analysis, the reader is referred to [6].

There is a simple extension of the above algorithm which produces information allowing us to determine the shortest path from s to arbitrary points x inside P. This extension partitions P into $O(n)$ triangular regions such that the shortest paths to s from all points in a region are identical except for a portion which consists of a straight segment from the point to one of the vertices of the region.

This extended algorithm depends on the following observations: For each edge e of P, consider $\Phi(e)$, the region bounded by e and the funnel F_e. First, note that for any point x inside $\Phi(e)$, we can find a vertex y of F_e such that either $\overline{xy}$ is tangent to

F_e, or y is the cusp of F_e and xy does not intersect F_e, as in the algorithm above. The shortest path from x to s then consists of the shortest path from y to s, concatenated with the segment $\overline{xy}$. Also note that the interiors of the $\Phi(e)$ are all disjoint, and the total number of edges along their boundaries is $O(n)$.

The extended algorithm consists of extending the edges of each funnel F_e to reach e, thereby partitioning $\Phi(e)$ into triangles; since each triangle corresponds to at least one funnel edge, there are $O(n)$ such triangles (see Figure 3.2). Then we can use any of the known linear-time algorithms (e.g., [9]) to process this partitioning into a structure supporting queries taking $O(\log n)$ time to determine into which triangle a point falls. Then of course the shortest path from any point x inside P to s can be constructed by determining the triangle into which x falls and concatenating the line from x to the appropriate vertex y of the triangle, as determined when the triangle was constructed, plus the path from y to s.

Summarizing, given a simple polygon P with n sides, and some vertex s of P, it is possible to preprocess P in linear plus triangulation time to obtain a data structure allowing us to determine, for any point x inside P, the length of the shortest path from x to s in $O(\log n)$ time. The path itself may be calculated in $O(\log n + k)$ time, where k is the number of segments in the path.

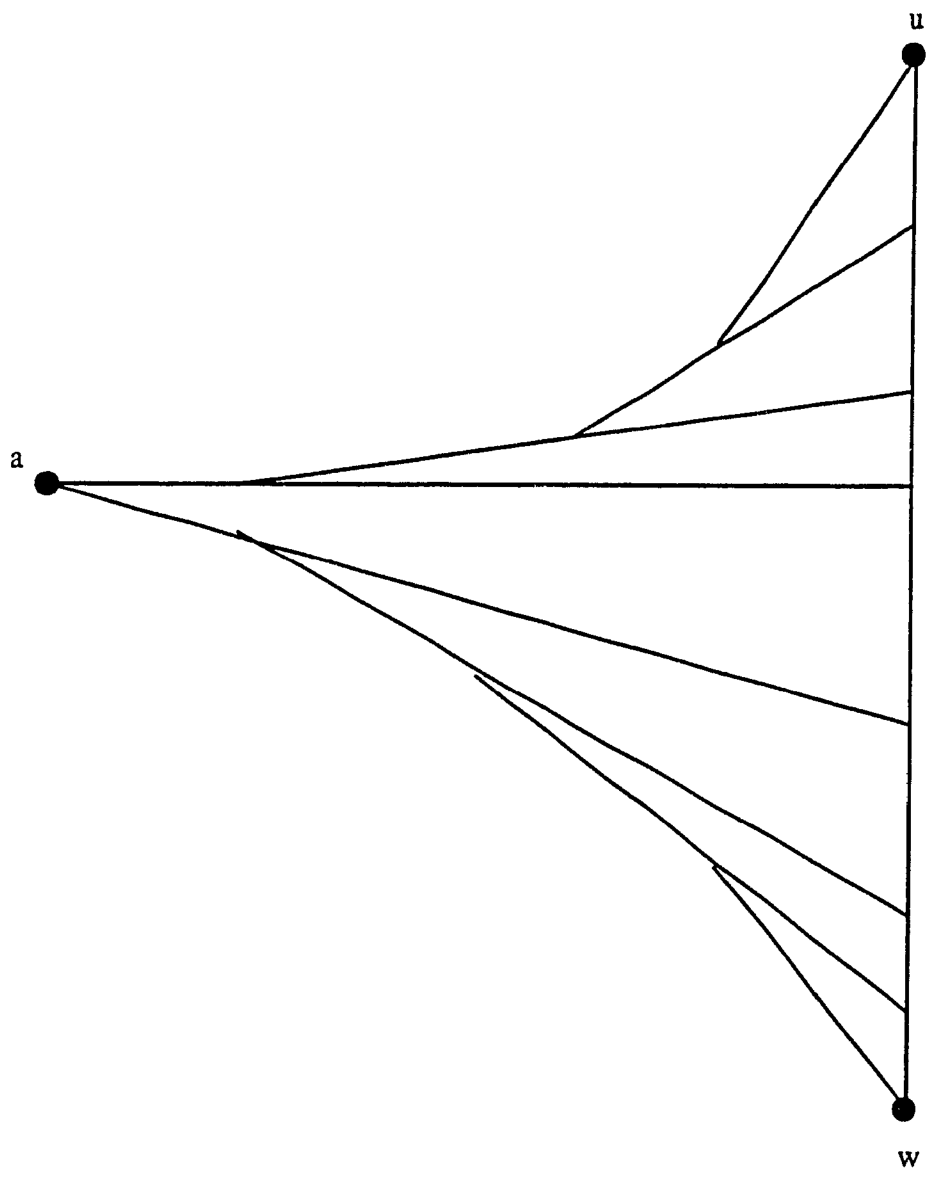


Figure 3.2: Partitioning a funnel into triangles

Chapter 4

Suri's Algorithm

In this section, we describe Suri's [17] $O(n \log n)$ method for computing the internal geodesic diameter of a simple polygon. Suri's method actually involves solving the *all geodesic furthest neighbours* problem and simply taking the maximum of the distances obtained.

More specifically, given a polygon P, the *all geodesic furthest neighbours* problem for P is to find, for each vertex v of P, a point $\phi(v) \in P$ such that

$$\delta_\iota(v, \phi(v)) = \max_x \delta_\iota(v, x)\, \forall x \in P.$$

In a manner similar to that of the previous chapter, this chapter will be devoted to a review of Suri's method for computing the internal geodesic diameter of a simple polygon. This method is presented here in order to lay the groundwork for the later chapters.

4.1 Observations

There are a few observations that must be made before proceeding to the algorithm; one is that the triangle inequality holds for geodesics (the proof of this is similar to that of Lemma 2.8). We also observe that the furthest neighbour of any point in P is always a vertex of P and that if P is triangulated, we can compute $\phi(u)$ in linear time for any given point u in P. In fact, this immediately gives us a brute force $O(n^2)$ algorithm to solve the all-geodesic-furthest neighbours problem. We simply have to compute the shortest path tree for each vertex. This, however, is not our ultimate goal.

We say that two geodesics are *disjoint* if they have no point in common. Let u, v, x, and y be four points on $bd(P)$ appearing in counterclockwise order (see Figure 4 1). The uniqueness of geodesics in a simple polygon insures that either $\pi_\iota(u,y)$ and $\pi_\iota(v,x)$ are disjoint or that they overlap only along a contiguous subpath. The geodesic between u and x can be broken down as follows

$$\pi_\iota(u,x) = \pi_\iota(u,a) \mid\mid \pi_\iota(a,c) \mid\mid \pi_\iota(c,x)$$

where $\mid\mid$ is concatenation of geodesics, a lies on $\pi_\iota(u,y)$, c lies on $\pi_\iota(v,x)$, and $\pi_\iota(a,c)$ is disjoint from both $\pi_\iota(u,y)$ and $\pi_\iota(v,x)$, save for the endpoints. Similarly we have

$$\pi_\iota(v,y) = \pi_\iota(v,b) \mid\mid \pi_\iota(b,d) \mid\mid \pi_\iota(d,y)$$

and the associated constraints.

If $\pi_\iota(u,y)$ and $\pi_\iota(v,x)$ are not disjoint, then we set $a = b = c = d = z$, for some arbitrary point z which is common to both geodesics. Otherwise, due to the uniqueness of geodesics in P, we have a, b, c, and d being all unique. The points a, b, c, and d are called the *junction points* for $\pi_\iota(u,y)$ and $\pi_\iota(v,x)$. This leads us to a simple lemma:

Lemma 4.1 *Given the situation described above, the following inequalities must hold:*

$$\delta_\iota(u,a) \leq \delta_\iota(u,d) \text{ and } \delta_\iota(v,b) \leq \delta_\iota(v,c).$$

Proof Obvious from Figure 4.1. □

The above lemma is used to prove the property described in the following lemma, known as the *crossing property.*

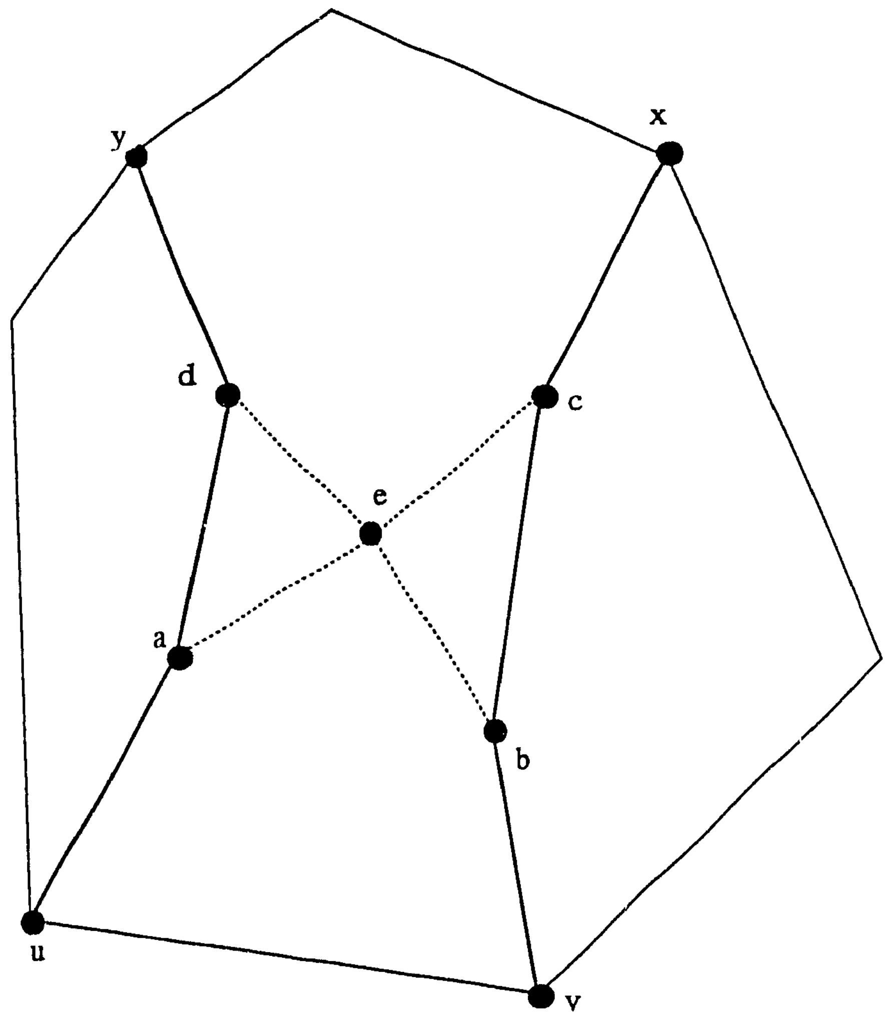


Figure 4.1: Junction points (also, the crossing property)

Lemma 4.2 *Let u, v, x, and y be four points on $bd(P)$ appearing in counterclockwise order such that x is a furthest neighbour of v and y is a furthest neighbour of u. Then, x is also a furthest neighbour of u and y is also a furthest neighbour of v.*

Proof By contradiction. By comparing $\delta_\iota(u,y)+\delta_\iota(v,x)$ with $\delta_\iota(u,x)+\delta_\iota(v,y)$, we see that if x is not a furthest neighbour of u, then the triangle inequality is violated. □

We will first describe Suri's solution for the *restricted furthest neighbours* problem and then show how to transform the original problem into at most three instances of the restricted problem.

4.2 Restricted Furthest Neighbours Problem

Two polygonal chains are said to be *non-overlapping* if they do not share a point, save possibly the endpoints. Let $U = (v_1, u_2, \ldots, u_s)$ and $V = (v_1, v_2, \ldots, v_t)$ be two non-overlapping chains of P, where the vertices of U are in counterclockwise order and those of V are in clockwise order. We denote $\zeta(u_i)$ as the vertex in V that is furthest from the vertex $u_i \in U$. In other words, $v_j = \zeta(u_i)$ is a *restricted furthest neighbour* of u_i if

$$\delta_\iota(u_i, v_j) = \max_{1\leq k\leq t} \delta_\iota(u_i, v_k)$$

The all-furthest neighbours problem for U with respect to V is to find the restricted furthest neighbour in V for each u_i. We will drop the word restricted for the rest of this section.

Let $P[a,b;c,d]$, for $1 \leq a \leq b \leq s$ and $1 \leq c \leq d \leq t$, denote the closed region in P whose boundary consists of $(u_a, \ldots, u_b)$ and $(v_c, \ldots, v_d)$ joined by $\pi_\iota(u_a, v_c)$ and $\pi_\iota(u_b, v_d)$. The aforementioned paths are called the *connectors* of $P[a,b;c,d]$, and we call their edges *connector-edges*. We can see that $\pi_\iota(u_i, v_j)$ lies completely in

$P[1,s;1,t]$, for $1 \leq i \leq s$ and $1 \leq j \leq t$. Therefore, we only have to consider $P[1,s;1,t]$ for a solution to our all-furthest neighbours problem for U with respect to V. Simply, we have to find $\zeta(u_i)$ for all $i = 1, 2, \ldots, s$.

Suri uses the following lemma to derive a divide-and-conquer algorithm.

Lemma 4.3 *Let $u_p \in U$ be a vertex and let $v_q = \zeta(u_p)$ be its furthest neighbour, for some $1 \leq p \leq s$ and $1 \leq q \leq t$. Then,*

1. *For every $u_i \in (u_1, u_2, \ldots, u_p)$, there exists a $v_k \in (v_q, v_{q+1}, \ldots, v_t)$ such that $v_k = \zeta(u_i)$.*
2. *For every $u_j \in (u_p, u_{p+1}, \ldots, u_s)$ such that $v_m \in (v_1, v_2, \ldots, v_q)$, there exists a $v_m = \zeta(u_j)$.*

Proof Follows from the previous lemma. □

The algorithm follows directly from the above lemma.

Algorithm *RFN*$(P[1,s;1,t])$

(1) If $|s-1| \leq 2$ or $|t-1| \leq 2$, then compute all furthest neighbours in linear time by constructing the shortest path trees from the vertices of the smaller chain.

(2) Otherwise, do the following:

(2a) Let $p = \left\lceil \frac{1}{2}s \right\rceil$ and let $v_q = \zeta(u_p), 1 \leq q \leq t$.

(2b) Construct $P[1,p;q,t]$ and $P[p,s;1,q]$.

(2c) Recursively call *RFN*$(P[1,p;q,t])$ and *RFN*$(P[p,s;1,q])$.

4.3 All Furthest Neighbours Problem

Let $(u_1, u_2, \ldots, u_n)$ be the vertices of P in counterclockwise order and assume that $u_{n+1} = u_1$. Let $(u_a, ..., u_b)$ denote that portion of $bd(P)$ encountered between u_a and u_b. Let u_i be an arbitrary vertex of P. We can assume, without loss of generality, that $u_j = \zeta(u_i)$ and $u_k = \zeta(u_j)$ are such that u_i, u_j, and u_k appear in this order on $bd(P)$, but u_i and u_k are not necessarily distinct. The next lemma shows how we can reduce the *all furthest neighbours* problem to at most three instances of the *restricted all furthest neighbours* problem for polygonal chains.

Lemma 4.4 *Let u_i, u_j, and u_k be as above. Then,*

1. *for every $u_l \in (u_i, \ldots, u_j)$, there always exists a $u_m \in (u_j, \ldots, u_k, \ldots, u_i)$ such that $u_m = \phi(u_l)$.*
2. *for every $u_l \in (u_j, \ldots, u_k)$, there always exists a $u_m \in (u_k, \ldots, u_i, \ldots, u_j)$ such that $u_m = \phi(u_l)$.*
3. *for every $u_l \in (u_k, \ldots, u_i)$, there always exists a $u_m \in (u_i, \ldots, u_j, \ldots, u_k)$ such that $u_m = \phi(u_l)$.*

Proof See [17]. □

Theorems 4.5 and 4.6 conclude our presentation of Suri's results.

Theorem 4.5 *Furthest neighbours of all the vertices of a simple polygon having n vertices can be computed in $O(n \log n)$ time and $O(n)$ space.*

And since $\delta_i(P) = \max_{u \in P} \delta_i(u, \phi(u))$, we get

Theorem 4.6 *The geodesic diameter of a simple polygon having n vertices can be computed in $O(n \log n)$ time and $O(n)$ space.*

Chapter 5

The External Geodesic Diameter Algorithm

This chapter presents the concepts of a depth profile and a restricted depth profile. We also present out algorithm for the external geodesic diameter and the complexity analysis of the algorithm. Finally, we present an algorithm for computing all the external geodesic furthest neighbours on P for all vertices of P.

5.1 Depth Profile

Let Q be a pocket of simple polygon P. Let a and b be the endpoints of Q, and let $l = \delta(a, b)$. Let q be any point of Q, and define

$$m(d, q) = \min(\delta_e(a, q), d + \delta_e(b, q)),$$

where d is a real-valued parameter.

The *depth profile* of Q is a mapping

$$\psi_Q : \Re \longmapsto [0, \infty) \text{ such that } \psi_Q(d) = \max_{q \in Q} m(d, q).$$

Associated with each point of the profile is the set of points which attain the maximum in the above definition:

$$\Psi_Q(d) = \{q \in Q | m(d, q) = \psi_Q(d)\}.$$

To see the relevance of these definitions, consider the following lemma:

Lemma 5.1 *Let $\Delta_\varepsilon(P)$ be an external geodesic diameter with endpoints $q_0 \in Q$ and $q_0' \notin Q$. Let S_a and S_b be the semi-circuits defined by q_0 and q_0' passing through a and b respectively, and let l_a and l_b be their lengths. Let $l_a' = l_a - \delta_\varepsilon(a, q_0)$ and $l_b' = l_b - \delta_\varepsilon(b, q_0)$. If $l_b' - l_a' = d$, then $\delta_\varepsilon(P) = l_a' + \psi_Q(d)$.*

Proof By definition,

$$\begin{aligned}
\delta_\varepsilon(P) &= \min\{l_a, l_b\} \\
&= \min\{l_a' + \delta_\varepsilon(a, q_0), l_b' + \delta_\varepsilon(b, q_0)\} \\
&= \min\{l_a' + \delta_\varepsilon(a, q_0), l_a' + d + \delta_\varepsilon(b, q_0)\} \\
&= l_a' + \min\{\delta_\varepsilon(a, q_0), d + \delta_\varepsilon(b, q_0)\} \\
&= l_a' + \psi_Q(d).
\end{aligned}$$

□

As a corollary of this lemma, any geodesic of the form $\pi_\varepsilon(q_0', q)$ for $q \in \Psi_Q(d)$ is also an external geodesic diameter of P. Thus in practice, given a point $q_0' \notin Q$, if the lengths l_a' and l_b' are known, the depth profile of Q allows one to determine an external geodesic furthest neighbour restricted to Q.

To calculate $\psi_Q(d)$ for any given value d, we will first preprocess the pocket Q. The first step is to compute the two shortest path trees rooted at a and b. Consider $(t_0, t_1, \ldots, t_s)$ to be all the nodes of the trees listed in sorted order along the boundary of Q from a to b, where $a = t_0$ and $b = t_s$. Let e_i be the line segment in Q with endpoints the consecutive nodes t_{i-1} and t_i. We shall consider the *restricted depth profile* $\psi_i'(d)$ to be defined as follows:

$$\psi_i' : \Re \longmapsto [0, \infty) \text{ such that } \psi_Q(d) = \max_{q \in e_i} m(d, q).$$

In terms of $\psi'_i(d)$, $\psi_Q(d)$ becomes $\max_{1 \le i \le s} \psi'_i(d)$.

To compute $\psi'_i(d)$ given a specific value of d, we note that if q_0 is not an endpoint of e_i, Lemma 2.5 implies that $\delta_\varepsilon(a, q_0) - \delta_\varepsilon(b, q_0) = d$. Let a'_i and b'_i be the ancestors of e_i in the shortest path trees rooted at a and b, respectively. Then $\delta_\varepsilon(a'_i, q_0) - \delta_\varepsilon(b'_i, q_0) = d - \delta_\varepsilon(a, a'_i) + \delta_\varepsilon(b, b'_i)$, which defines a hyperbola.

Algorithm *RDP*(d)

The algorithm assumes that the shortest path trees rooted at a and b are available, and that Q has been partitioned into contiguous line segments as decribed above.

(1) Let a'_i and b'_i be the ancestors of e_i in the shortest path trees rooted at a and b, respectively.

(2) Transform the line segment e_i to coincide with the x-axis of a new coordinate system. Transform the ancestors described in Step 1 into this new system.

(3) Compute the points of intersection of the hyperbola with the x-axis. Discard those points not lying on the transformed image of e_i. Find those points of e_i corresponding to the remaining intersection points.

(4) Of these points together with the endpoints of e_i, return the point x_0 maximizing the distance $\psi'_i(d) = \min\{\delta_\varepsilon(a, x), d + \delta_\varepsilon(b, x)\}$, together with $\psi'_i(d)$.

Since at most a linear number of extra nodes is generated in the construction of the shortest path trees, $s = O(k)$, where k is the number of edges in pocket Q [6]. In terms of k, the preprocessing for Algorithm *RDP* requires $O(k \log k)$ time [6]. The merging of the two trees requires time linear in k. Step 1 can be performed in constant time providing that back pointers from the partition of Q to the leaves of the shortest path trees are maintained. The transformation in Step 2 requires constant time assuming that the trigonometric functions are considered as constant time operations. Steps 3 and 4 are easily seen to require only constant time.

Thus to compute $\psi_Q(d)$, it suffices to apply Algorithm *RDP* to all line segments e_i of the partition of Q, and to retain that point q_0 at which $\max_{1 \le i \le s} \psi_i'(d)$ is attained. These arguments establish the following lemma:

Lemma 5.2 *Given the pocket Q preprocessed as in the description of Algorithm RDP, the depth profile $\psi_Q(d)$ may be evaluated for a given d in $O(k)$ time, where k is the number of edges of Q.*

5.2 The Algorithm and its Analysis

Algorithm EGD

Input: Simple polygon P.

Output: A maximal external geodesic path for P.

1. Determine the convex hull of P.
2. Determine the number of pockets of P.
   (a) If P has no pockets (P is convex), then
      i. Find the perimeter length of P.
      ii. From any vertex find the point one-half the perimeter length away.
      iii. These two points define an external geodesic diameter of P.
   (b) If P has exactly one pocket, then
      i. Find the perimeter length of P, less the length of the lid of the pocket.

ii. Compute the shortest path trees of the pocket rooted at the endpoints of its lid.

iii. Find the internal geodesic diameter of the pocket polygon.

iv. Find the vertex v which maximizes

$$\delta_\iota(x, v) + \delta_\iota(y, v)$$

where x and y are the endpoints of the lid.

v. Find the corresponding point, w, to v on the exterior of the original polygon, P.

vi. Take the maximum of $\delta_\iota(v, w)$ and the result of step 2(b)iii.

vii. The result of step 2(b)vi is an external geodesic diameter of P.

(c) If P has two or more pockets, then

i. For each pocket of P do step 2b.

ii. For each pocket of P preprocess in preparation for Algorithm *RDP*.

iii. For each pocket vertex, find the deepest point in each of the other pockets by applying Algorithm *RDP*.

iv. For each pocket, take the maximum of step 2(c)i and step 2(c)iii.

v. The maximum of step 2(c)iv over all pockets is an external geodesic diameter of P.

5.3 Analysis

The proof of correctness of Algorithm *EGD* follows from the previous results. The complexity analysis of the algorithm follows:

Case 1: $O(n)$ by one of the various convex hull algorithms.

Case 2a: $O(n)$ trivially.

Case 2b: Step 2(b)i requires $O(h)$ time, where h is the number of convex hull vertices. However, applying Guibas *et al.*'s [6] algorithm for creating shortest path trees requires $O(k \log k)$ time, where k is the number of vertices in the pocket. Suri's [17] algorithm for computing the internal geodesic diameter of a simple polygon also requires $O(k \log k)$ time. Step 2(b)iv requires k queries of $O(\log k)$ time each, which gives $O(k \log k)$ time, while Step 2(b)v needs $O(h)$ time. This gives us a time complexity of $O(k \log k)$, which is worstcase $O(n \log n)$.

Case 2c: Steps 2(c)i and 2(c)ii each require $O(k \log k)$ time for each pocket, which yields a total time complexity of $O(n \log n)$. Step 2(c)iii requires $O(k)$ for each pocket. This gives a time complexity of $O(n)$ for each vertex, and a total time complexity of $O(n^2)$. Step 2(c)iv requires time $O(m)$ where m is the number of pockets of P.

Due to the limiting factor of Step 2(c)iii, the algorithm has a time complexity of $O(n^2)$.

5.4 The All External Furthest Neighbours Algorithm

This algorithm is a simple extension of Algorithm *EGD*, and has a similar complexity analysis. It computes and outputs the external geodesic furthest neighbours for every vertex of P.

Algorithm AEGFN

Input: Simple polygon P.

Output: $\phi_e(v) \, \forall v \in P$

1. Determine the convex hull of P.

2. Determine the number of pockets of P.

   (a) If P has no pockets (P is convex), then

      i. Find the perimeter length of P.

      ii. For each vertex, find the corresponding point one-half the perimeter length away.

   (b) If P has exactly one pocket, then

      i. Find the perimeter length of P, less the length of the lid of the pocket.

      ii. Create the shortest path map for the pocket polygon.

      iii. Apply Suri's *all geodesic furthest neighbours* algorithm to the pocket polygon.

      iv. For each pocket vertex, find its furthest-neighbour outside the pocket.

      v. For each pocket vertex, take the maximum of the results of step 2(b)iii and step 2(b)iv.

      vi. For each pocket of P preprocess in preparation for Algorithm *RDP*.

      vii. For each vertex outside the pocket, find the corresponding deepest point inside the pocket.

      viii. For each vertex outside the pocket, find the corresponding point one-half the perimeter length away.

      ix. For each vertex outside the pocket, take the maximum of the results of step 2(b)vii and step 2(b)viii.

   (c) If P has two or more pockets, then

      i. For each pocket of P do step 2b.

      ii. For each pocket vertex, find the deepest point in each of the other pockets.

iii. For each pocket, take the maximum of step 2(c)i and step 2(c)ii.

Chapter 6

Conclusion

In this thesis we have presented concepts about geodesics and their behavior in and on simple polygons. We have also presented algorithms for computing the external geodesic diameter of a simple polygon and for solving the *all external geodesic furthest neighbours* problem. There are a few open problems that are immediately suggested by our results:

- reducing the time complexity of the algorithms. This is promising since the time complexity is established by only one step of the algorithms. If some method could be found to reduce the run time of Algorithm *RDP*, perhaps by preprocessing the pocket so as to generate a complete depth profile, then the algorithms would run more efficiently.
- finding non-trivial lower bounds.
- extending these results to higher dimensions.

Bibliography

[1] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility-Polygon Search and Euclidean Shortest Paths. *Proc. of 26th IEEE Symp. on Foundations of Computer Science,* 1985, pp. 155–164.

[2] T. Asano and G. Toussaint. Computing the Geodesic Center of a Simple Polygon in *Discrete Algorithms and Complexity, Proc. of the Japan-U.S. Joint Seminar,* June 4–6, 1986, Kyoto, Japan, pp. 65–79.

[3] B. Chazelle. A Theorem on Polygon Cutting with Applications. *Proc. of 23rd IEEE Symp. on Foundations of Computer Science,* 1982, pp. 339–349.

[4] H. Davenport and A. Schinzel. A Combinatorial Problem Connected with Differential Equations. *Amer. J. Math.,* **87**, 1965, pp. 684–694.

[5] L. Guibas and J. Hershberger. Optimal Shortest Path Queries in a Simple Polygon. *Proc. of the Third Ann. Symp. on Comp. Geom.,* 1987, pp. 64–75.

[6] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear Time Algorithms for Visibility and Shortest Path Problems Inside Simple Polygons. *Proc. of the Second Ann. Symp. on Comp. Geom.,* 1986, pp. 1–13.

[7] L. Guibas, E. McCreight, M. Plass, and J. Roberts. A New Representation for Linear Lists. *Proc. 9th ACM Symp. on Theory of Computing,* 1977, pp. 49–60.

[8] S. Huddleston and K. Mehlhorn. A New Data Structure for Representing Sorted Lists. *Acta Informatica,* **17**, 1982, pp. 157–184.

[9] D. Kirkpatrick. Optimal Search in Planar Subdivisions. *SIAM J. Computing*, **12**, 1983, pp. 28–35.

[10] D. T. Lee and F. P. Preparata. Euclidean Shortest Paths in the Presence of Rectilinear Barriers. *Networks*, vol. 14, **3**, 1984, pp. 393–410.

[11] D. Mount. On Finding Shortest Paths on Convex Polyhedra. Tech. Rept., University of Maryland, May 1985.

[12] R. Pollack, G. Rote and M. Sharir. Computing the Geodesic Center of a Simple Polygon. Tech. Rept. 231, Computer Science Department, Courant Institute, July 1986.

[13] R. Pollack and M. Sharir. Computing the Geodesic Center of a Simple Polygon in "Research Workshop on Moveable Separability of Sets," G. Toussaint, ed., Bellairs Research Institute of McGill University, February 1986, pp. 26–47.

[14] J. Reif and J. Storer. Shortest Paths in Euclidean Space with Polyhedral Obstacles. Tech. Rept., Brandeis University, April 1985.

[15] J. Reif and J. Storer. Minimizing Turns for Discrete Movement in the Interior of a Polygon. Tech. Rept., Harvard University, December 1985.

[16] M. I. Shamos. Problems in Computational Geometry. PhD thesis, Carnegie-Mellon University, 1977.

[17] M. Sharir and A. Schorr. On Shortest Paths in Polyhedral Spaces. *Proc. 16th ACM Symp. on Theory of Computing,* 1984, pp. 144–153.

[18] S. Suri. The All-Geodesic-Furthest Neighbours Problem for Simple Polygons. *Proc. of the Third Ann. Symp. on Comp. Geom.,* 1987, pp. 64–75.

[19] E. Szemeredi. On a Problem by Davenport and Schinzel. *Acta Arith.,* **25**, 1974, pp. 213-224.

[20] G. Toussaint. Computing Geodesic Properties of Polygons, manuscript in preparation.