Visibility in Simple Polygons

by

Prosenjit K. Bose

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 1991

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Acknowledgements

No words can express my gratitude to both my supervisors, Ian Munro and Anna Lubiw. I thank them for their ideas, suggestions, guidance, constant encouragement, support, and above all, their friendship. I want to thank Ian for convincing me to pursue graduate studies and instilling in me the enthusiasm he has for research. I thank Anna for introducing me to Computational Geometry, and for always having faith in me even during the times when I lost faith in myself. This thesis would not have been possible without them.

I thank my readers, Prabhakar Ragde and Derick Wood, for their critical reading of this thesis and their valuable comments. I also want to thank Prabhakar for his friendship and for always pushing me beyond my limit in every course I took with him.

I would like to thank Subhash Suri for suggesting the investigation of line segment queries.

Of the many good friends I have made here: I would like to thank Naji Mouawad for the many interesting discussions we had. I thank Tom Papadakis, Shiva Chaudhuri, and Paul Colley for carefully reading early drafts of this thesis. I also thank Venkatesh Raman, Dave Wagner, Murray Sherk, and Naomi Nishimura.

I thank my mother, father and sister for their love and support. I thank Shawna for putting up with me and my moods.

Finally, the financial support from the Department of Computer Science, Information Technology Research Centre, Institute for Computer Research and the Natural Sciences and Engineering Research Council is gratefully acknowledged.

Dedication

This thesis is dedicated to my parents on the occasion of their 25th wedding anniversary. Without their years of hard work, love, support and endless encouragement, it would not exist.

Prosenjit K. Bose
December 9, 1991

Abstract

We present a new method of decomposing a simple polygon which allows the preprocessing of the polygon to efficiently answer queries of various forms. Using $O(n^3 \log n)$ preprocessing time and $O(n^3)$ space, we can, given a query point q inside or outside the polygon, recover the number of vertices visible from q in $O(\log n)$ time. Also, we can recover the visibility polygon of q in $O(\log n + k)$ time, where k is the size of the visibility polygon.

The key notion behind the decomposition is the succinct representation of *visibility regions*. We show that there can be $\Theta(n^3)$ distinct visibility regions, each of which can see $\Theta(n)$ points. The main technical theorem is that there can be $\Theta(n^2)$ regions with *minimal visibility sets*, thus permitting preprocessing in $O(n^3 \log n)$ time and $O(n^3)$ space.

These techniques are extended to handle other types of queries. Given m fixed points in the plane, and a polygon of size n, we can, given a query point q inside or outside the polygon, recover the number of fixed points unobstructed by the polygon boundary in $O(\log mn)$ time and recover the visible set of size k in $O(\log mn + k)$, with $O(m^2(m+n)\log n)$ preprocessing time and $O(m^2(m+n))$ space.

We also explore line segment queries. Given a line segment inside or outside a polygon of size n, we can recover its weak visibility set and more generally its weak visibility sequence in $O(k \log n)$, using $O(n^3 \log n)$ preprocessing time and $O(n^3)$ space. To answer weak visibility line segment queries on a set of m fixed points in the plane, we need $O(k \log mn)$ query time, given $O(m^2(m+n)\log n)$ preprocessing time and $O(m^2(m+n))$ space.

Contents

List of Figures

Chapter 1

Introduction

The notion of visibility among geometric objects is one of the most fundamental topics in computational geometry. Many different visibility problems have been studied in the literature. The ray shooting problem (where, given a polygon and a ray emanating from a given point in a given direction, the object is to find the first intersection, if any, of the ray with the polygon boundary) has been studied by Chazelle [7], Chazelle and Guibas [9], and Guibas et al [17]. The guarding problem (where, given a polygon, the object is to station guards at the vertices in order to cover the whole polygon) and its many variations has been studied by Chvátal [12], Aggarwal [1], O'Rourke [26], and Shermer [30]. The visibility polygon problem (where, given a polygon P and a point q inside the polygon, determine the polygon consisting of all the points in P which are visible from q) has been studied by ElGindy and Avis [16], Lee [22], Joe and Simpson [20]. O'Rourke's book [26] surveys many visibility problems which have been investigated.

In this thesis, we consider several questions concerning visibility, all stemming from the following question: Given a query point q in the interior of a simple polygon, P, find all the *vertices* of P that *see* q. The notion of points seeing other points in a polygon is formally defined in Chapter 2. The single-shot version of this problem is a special case of the visibility polygon problem and it can be solved in $O(n)$ time where n is the number of vertices in the polygon. In this thesis, we develop a data structure that can be used to answer repeated queries efficiently. After the data structure has been built, the query algorithm can find the number of vertices of P which see q in $O(\log n)$ time, where n is the number of vertices of P. Listing the vertices takes $O(\log n + k)$ time, where k is the number of vertices.

The succinctness of the data structure is based on some properties of the polygon decomposition into *visibility regions*. Several key properties of this polygonal decomposition method allow us to solve extensions of the original problem, as well as some different problems, quite efficiently.

We uncover some properties of the decomposition which allow us to recover the whole visibility polygon of the query point as opposed to just the visible vertices. The visibility polygon represents all the points in the polygon that see the query point. Informally, we can think of the query point q as a light source in a dark art gallery, represented by P, and we want to know what area of the gallery is lit by q. As this thesis was being completed, we learned that the notion of visibility regions and how to use them to handle visibility polygon queries was being discovered simultaneously and independently by Guibas et al. [18].

We generalize from visibility of vertices to a set of fixed points of interest. Given a fixed set of points, S, in the plane, a polygon P, and a query point q, we are interested in the subset of S seen from q, unobstructed by the boundary of P. Informally, we can think of the points of S as light sources in an art gallery, represented by P, and we wish to know which lights are shining on an observer, q, in the gallery.

We are able to extend the structure to deal with query points either inside or outside a simple polygon P, and solve all of the above situations. An alternative approach using persistent data structures and visibility regions is explored.

The visibility decomposition structure reveals much geometric information about the polygon. We study some applications of the visibility decomposition with respect to line segment queries. In particular, we look at weak visibility queries with respect to line segments as opposed to strong visibility. A point is weakly visible from a line segment if it sees any part of the line segment, and it is strongly visible from the line segment, if it sees the whole line segment.

1.1 Overview

We will now give an intuitive notion of the idea used to preprocess the polygon for efficiently answering visibility queries. A *visibility region*, R, in a simple polygon P, is a *maximally connected* subset of P with the property that any two points in R see the same subset of vertices of P. The *visibility set* of a visibility region, R, is the subset of vertices of the polygon P which can be seen from R. Two visibility regions are *neighboring* visibility regions if they share a common edge. A visibility region with *minimal visibility set* is one whose neighboring visibility regions have visibility sets which strictly contain it. Such regions will be referred to as *sinks* and the analogy to the standard definition of sinks in digraphs will soon become clear.

In order to give a compact representation of the decomposition of a polygon into visibility regions, we take advantage of several key properties of the decomposition. First, using a counting argument, we show that a simple polygon P on n vertices has $O(n^3)$ visibility regions. This immediately offers a solution to the query problem. We simply associate with each region the subset of vertices it sees. When given a query point, we determine, using standard planar point location techniques, which visibility region it is in. However, it is possible for every region to see a linear number of vertices which would result in $\Theta(n^4)$ space for the structure.

To reduce the space requirements, we consider the dual graph of the planar map of visibility regions. We show that the visibility set of any two neighboring visibility regions differs by only one vertex; we direct the corresponding edge of the dual graph towards the visibility region with the smaller visibility set and label the edge by the lost vertex. We obtain a directed acyclic graph whose sinks correspond exactly to visibility regions with minimal visibility set. From any node representing a visibility region r, we can find a directed path π to a sink s; the visibility set of r is the visibility set of s plus the set of vertices labelling the edges of π. This notion was first used by Chazelle [8] in a different context.

Through the use of a technical counting argument, we are able to establish that there are $O(n^2)$ sinks or regions with a minimal visibility set. By associating with each such region the subset of vertices it sees, we are able to store the information necessary for answering the queries efficiently, in $O(n^3)$ space. See Figure 1.1.

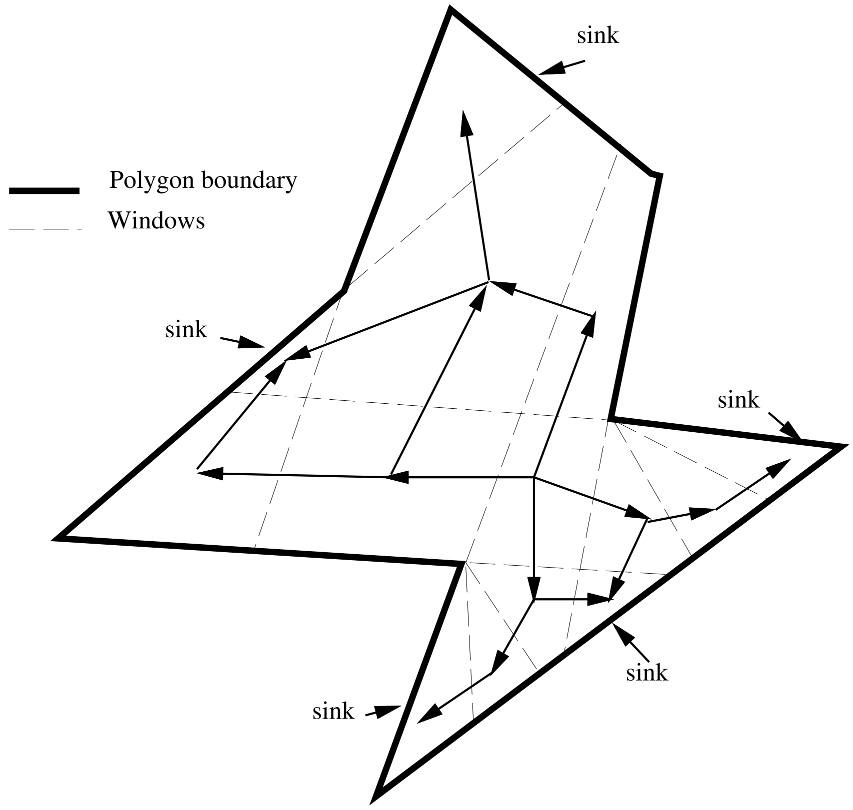

Figure 1.1: Polygon decomposed into visibility regions with directed dual graph.

During the preprocessing phase, the polygon is decomposed into visibility regions. The regions are then preprocessed for planar point location (Kirkpatrick [21], Lee and Preparata [24], Preparata [27], and Sarnak and Tarjan [29]). The directed dual planar graph of the planar subdivision is built, where only the subset of vertices of P seen by each sink is stored. At query time, a standard $O(\log n)$ planar point location query reveals which region contains the query point. Once the region is located, we enumerate the vertices by following a path from the node to a sink. The time required for preprocessing is $O(n^3 \log n)$ since the cost is dominated by the time required to preprocess for planar point location. The space requirement is $O(n^3)$, which is optimal with respect to the decomposition.

All extensions forged from this structure follow naturally from the initial idea.

Chapter 2

Notation and Preliminaries

Most of the geometric and graph theoretic terminology used is standard and for details, we refer the reader to O'Rourke [26], Bondy and Murty [6], Preparata and Shamos [28]. For simplicity of presentation, we assume that no three vertices of the polygon are collinear. We will begin by reviewing some of the main geometric and graph theoretic terminology used in this thesis.

2.1 Geometric Terminology

Let E^2 denote the Euclidean space of dimension 2. A domain D in E^2 is *convex* if, for any two points q_1 and q_2 in D, the segment $\overline{q_1 q_2}$ is entirely contained in D. A *simple polygon* P is a simply connected subset of the plane whose boundary is a closed chain of line segments. As we are dealing only with simple polygons, we will refer to them as polygons in the remainder of the thesis. We will denote a polygon, P, by a set of vertices $v_1, v_2, \ldots, v_{n-1}, v_n$ such that each pair of consecutive vertices is joined by an edge, including the pair $\{v_n, v_1\}$. We assume that the points are in clockwise order, so that the interior of the polygon lies to the right as the boundary of the polygon is traversed. We will denote the open interior of the polygon P by $INT(P)$, the boundary by $BND(P)$, and the open exterior by $EXT(P)$. The boundary is considered part of the polygon; that is, $P = INT(P) \cup BND(P)$. Between any two points, s and t, on the boundary P, there is a clockwise polygonal chain from s to t, where the interior lies to the right of the chain and a counterclockwise chain where the interior lies to the left of the chain. We will denote the clockwise chain by $CWC(s,t)$ and the counter-clockwise chain by $CCWC(s,t)$. Clearly, $CWC(s,t) \cup CCWC(s,t) = BND(P)$.

We say a point p is *in* P when $p \in INT(P) \cup BND(P)$. Two points in a polygon *see* each other if the line segment between them does not intersect the exterior of the polygon. Note that the line segment may touch the boundary of the polygon at one or more points; that is, line-of-sight is not blocked by grazing contact with the boundary. There is a parallel notion of *exterior visibility.* When considering exterior visibility, we say a point p is in the *exterior* of P to mean $p \in EXT(P) \cup BND(P)$. Two points in the *exterior* of a polygon *see* each other if the line segment between them does not intersect the open interior of the polygon.

Let x be a point in polygon P. The *visibility polygon* from x, denoted by $VP(x,P)$, is the set of points in P visible from x; it is formally defined by

$$VP(x,P) = \{z \mid z \in P \text{ and } \overline{xz} \cap P = \overline{xz}.\}$$

An edge of $VP(x,P)$ which is not contained in an edge of P is called a *window* of point x. Of the two vertices of a window, the one closest to x is called a *base* and the other is called an *end.* Note that the base of a window is a vertex of the polygon P, but the end may not be. See Figure 2.2. A window is denoted by $Wind(base, end)$. We refine the definition of windows slightly to eliminate ambiguities that exist due to collinearities. If there is situation similar to that of (a) and (b) of Figure 2.1, where a point x is collinear with two reflex vertices of the polygon, we say that $Wind(b,e)$ is the window of point x. If a situation similar to that in (c) arises, we say that there are two windows, one is $Wind(b_1,e_1)$ and the other is $Wind(b_2,e_1)$. A *pocket* of x in a polygon P is a maximal connected subset of $P \backslash VP(x,P) \cup \{$ the connecting window$\}$. Each pocket is connected by a window to the visibility polygon and is denoted by $Pocket(base, end)$ with the base and end being that of the connecting window. A pocket is called a *left pocket* if, in the neighborhood of its

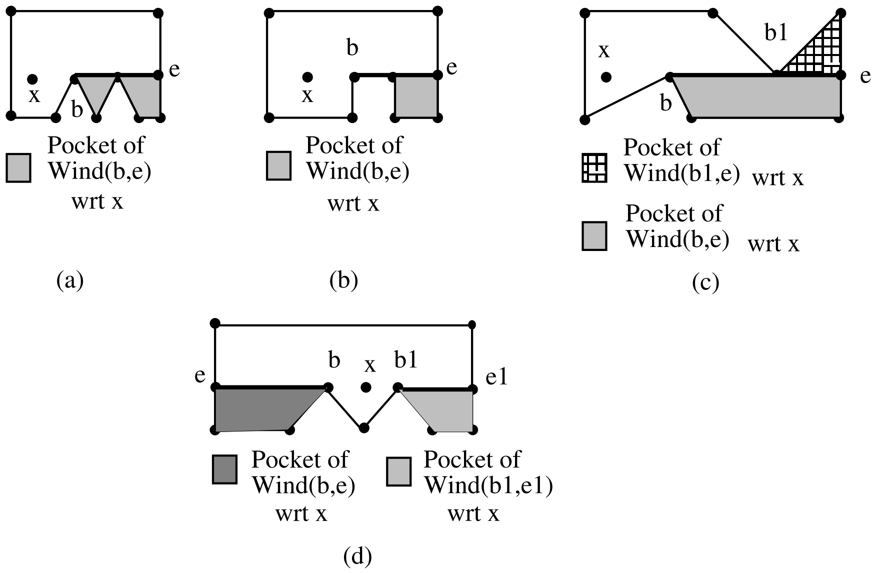

Figure 2.1: Refined definition of windows

window, the pocket lies to the left (counterclockwise) of the ray from x containing the window, and the visibility polygon to the right (clockwise). A similar definition holds for *right pockets*. A vertex v of P is called a *pocket vertex* with respect to $VP(x,P)$ if v is on a pocket. We extend this notion to *right pocket vertices* and *left pocket vertices* in the obvious way. A window bounding a left pocket or right pocket is called a *left window* or a *right window*, respectively. See Figure 2.2.

Lemma 2.1 *Let $Q = Pocket(b,e)$ of point x in polygon P. Let R be the half-plane, defined by the line through the line segment $\overline{be}$, in the direction of the pocket with respect to the window. If a point y is exterior to the region enclosed by Q, but contained in the half-plane R, then no point in the region enclosed by Q is visible from y. The half-plane R is known as a* window half-plane.

Proof: Let z be any point interior to the region enclosed by Q. Then the line segment $\overline{zy}$ intersects Q at least once. As $\overline{yz}$ cannot intersect $\overline{be}$, it intersects some

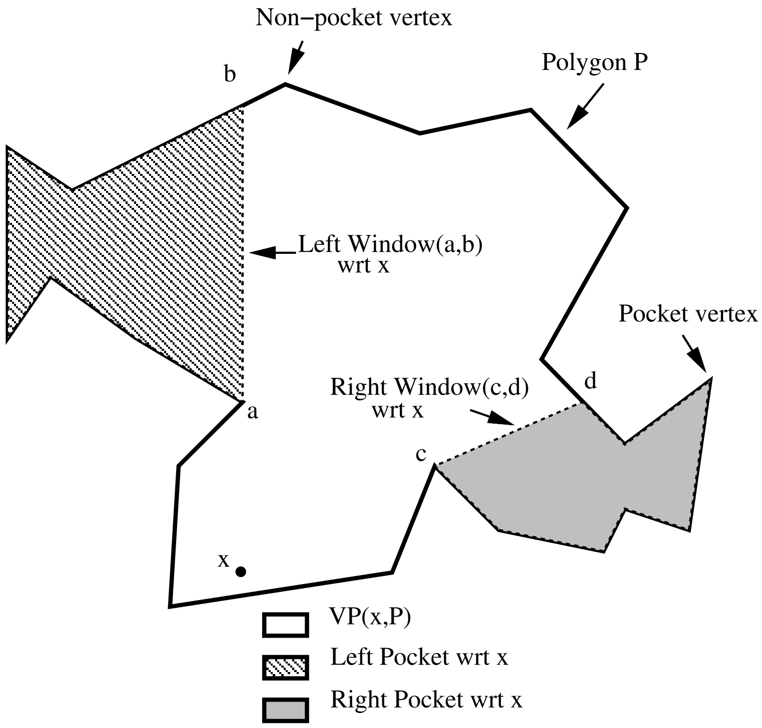

Figure 2.2: Windows and Pockets

other line segment on the boundary of Q. So, y and z are not visible with respect to the chain Q and similarly all points in that region. ■

2.2 Graph Theoretic Terminology

A *graph* $G = (V(G), E(G))$ consists of a finite nonempty set $V(G)$ of *vertices*, and a set $E(G)$ of unordered pair of vertices known as *edges*. A graph is *directed*, or known as a *digraph*, if the set $E(G)$ consists of ordered pairs of vertices known as *arcs*. If a is an arc and u and v are vertices such that $uv \in E(G)$, then a is said to *join* u to v; u is the *tail* of a, and v is its *head*.

A graph G is said to be a *planar graph* if it can be drawn in the plane so that its edges intersect only at their ends. Such a drawing of a graph is known as a *planar embedding.*

A planar embedding of a planar graph G partitions the plane into a number of connected regions known as *faces.* Each planar graph has exactly one unbounded face, called the *exterior face* or *external face.* All other faces are *non-external* faces.

A straight line planar embedding of a planar graph determines a partition of the plane known as a *planar subdivision* or *map.*

The planar *dual*, G^*, of a planar graph G embedded in the plane, is defined as follows: corresponding to each face f of G there is a vertex f^* of G^*, and corresponding to each edge e of G there is an edge e^* of G^*; two vertices f^* and g^* are joined by an edge e^* in G^* if and only if their corresponding faces f and g are separated by the edge e in G.

A *path* in a directed graph, G, is a finite non-null sequence $P = v_1 e_1 v_2 e_2 \ldots e_k v_k$ where the arcs $e_1, e_2 \ldots e_k$ are distinct, the vertices $v_1, v_2 \ldots v_k$ are distinct and v_i is the tail and v_{i+1} is the head of arc e_i in digraph G. The vertices v_1 and v_k are known as the *origin* and the *terminus* of the path. A *cycle* is a path whose origin and terminus are the same. An *acyclic* digraph is a digraph that contains no cycles.

Chapter 3

Visibility Decomposition and its Properties

In this chapter, we prove some key properties of windows and visibility regions which allow us to efficiently store the information necessary for the preprocessing step. Recall from the overview, in Chapter 1, that in the preprocessing phase, we decompose a polygon into visibility regions. If we let W_i be the set of all windows of a vertex v_i in polygon P, then we show that the non-external faces of the planar subdivision induced by $W = BND(P) \cup \bigcup_{i=1}^{n} W_i$ is precisely the set of visibility regions.

To succinctly store the information contained in the planar map of visibility regions, we show that there are $O(n^3)$ visibility regions and $O(n^2)$ regions of minimal visibility or *sinks* in a polygon. We also show that these bounds are tight by showing there exist polygons that have $\Theta(n^3)$ visibility regions and $\Theta(n^2)$ sinks.

We begin by studying the properties of windows in polygons that allow us to give a bound on the number of windows in the set W.

Lemma 3.1 *Given a window with respect to point x in polygon P, the base of the* *window is a reflex vertex of the polygon P.*

Proof: We noted that the base of a window is a vertex, so we need to show that it is a reflex vertex. By definition, the point x is collinear with the window. Let v be the base of the window and e be the end. Let a be the vertex on the polygon boundary occurring before v and b be the vertex occurring after v. Suppose that v is not a reflex vertex then the interior angle$(avb) < 180°$, which implies that the line segment $\overline{xe}$ intersects $EXT(P)$. ■

Since all windows emanate from reflex vertices, we make the following definitions. Let v be a reflex vertex in polygon P with a being the vertex occurring after v in the clockwise order of the vertices and b the vertex occurring before v. The *primary left window* with base v is the left window of b through v and the *primary right window* with base v is the right window of a through v.

All of the windows of a point x in a polygon P must be collinear with x. Thus, we have:

Lemma 3.2 *Given a point x in polygon P, each reflex vertex of the polygon can be the base of at most one window with respect to x.*

A polygon can have $O(n)$ reflex vertices; thus, the set W contains $O(n^2)$ windows.

We now give a more formal definition of the set W, and describe how to obtain a decomposition of a polygon into visibility regions. Given a polygon P, let W_i be the set of all windows in P with respect to vertex v_i. Let

$$W = BND(P) \cup \bigcup_{i=1}^{n} W_i$$

Consider the planar subdivision induced by all of the line segments in W. We refine the definition of a non-external face to appropriately include the edges of the planar subdivision. Two neighboring non-external faces are separated by an edge which is part of a window of some polygon vertex. The edge separating the two faces belongs to the face which is not in the pocket of that window. To avoid dealing with the degenerate situation where a visibility region is reduced to a single point, we will assume that no three windows emanating from three different reflex vertices intersect at a single point. Thus, we may consider the intersection point between windows to belong to the non-external face of the subdivision that is not in the pocket of any of the windows.

Lemma 3.3 *The set of non-external faces in a subdivision induced by all of the line segments in the set W is precisely the set of visibility regions.*

Proof: First, we show that an arbitrary non-external face f in such a subdivision is contained in a visibility region, r, of the polygon. We know that a face in a planar subdivision is a connected region. Thus, if f was not contained in a visibility region, then there must exist two points in f that do not see the same subset of vertices of the polygon. Let x and y be two such points. Without loss of generality, let v be a vertex of the polygon which is seen by x but not y. This implies that y lies in a pocket with respect to v. Therefore, since y is in a pocket with respect to v and x is not, they cannot lie on the same face.

Now we show that an arbitrary visibility region, r, of the polygon is contained in a non-external face, f. Suppose that a visibility region r was not a non-external face of such a subdivision. Since a visibility region is connected, there must exist two points in r that lie in two adjacent non-external faces. Let x and y be two such points, lying in faces f_1 and f_2 respectively. The faces f_1 and f_2 are separated by an edge which is part of some window. Let that window be a window with respect to vertex v. But this implies that either x or y does not see v. ■

Corollary 3.4 *Given two visibility regions that share a common edge, their visibility set is the same set except for one vertex.*

Lemma 3.5 *A visibility region R in a simple polygon P is convex.*

Proof: Because of Lemma 3.3, we need to show that, if two points are in a face of a planar subdivision into visibility regions, then the line segment between them is contained in the face.

Let a and b be two points in a face f. If a does not see b, then clearly they cannot be in the same face since any face is a visibility region. So, let us assume that a sees b. This means that $\overline{ab}$ is contained in the polygon. Suppose that $\overline{ab}$ is not contained in f, then there must be some window in the planar subdivision of visibility regions which intersects $\overline{ab}$. But, this contradicts the fact that a and b are in f. ■

A planar subdivision induced by n^2 arbitrary line segments in the plane can have $O(n^4)$ faces. The following two lemmas reveal the restrictions imposed by the notion of visibility in polygons. These restrictions are crucial in showing that there are only $O(n^3)$ faces in the planar subdivision of visibility regions induced by n^2 line segments (the windows).

Lemma 3.6 *Given a line segment $\overline{ab}$ inside polygon P, if a point z in P sees a and b then z sees any point on the line segment $\overline{ab}$.*

Proof: By the definition of visibility, the line segments $\overline{za}$ and $\overline{zb}$ lie completely inside P. Since triangle(z, a, b) lies inside P, we can conclude that z sees every point on the line segment $\overline{ab}$. ■

Lemma 3.7 *Given an arbitrary line segment inside a polygon P, and a point x inside P, at most two windows of x intersect the line segment.*

Proof: A window with respect to a point x forms the border between an area that is seen by x and one that is hidden. The result follows by Lemma 3.6. ■

Theorem 3.8 *There are $O(n^3)$ visibility regions in a polygon P on n vertices and there are polygons on n vertices that have $\Theta(n^3)$ visibility regions.*

Proof: By Lemma 3.2, we have

$$\|W\| \leq n^2.$$

Let t be the number of intersections on a single window of vertex v_i with all other windows from the set $\bigcup_{j \neq i} W_j$. By Lemma 3.7, we infer that $t < 2n$. Therefore, the total number of intersections in a decomposition is less than $2n^3$. Euler's Formula ([6], p. 143) states that if G is a connected plane graph, then $v - e + f = 2$, where v is the number of vertices, e is the number of edges, and f is the number of faces of G. We also know that in a planar graph, the relation $e \leq 3v - 6$ holds. Thus, there are no more than $4n^3$ visibility regions.

The polygon in Figure 3.1, is a polygon with $\Theta(n^3)$ visibility regions. Consider the visibility regions in the triangle(a, b, c) or spike. At least $\frac{n}{6}$ vertices form windows with base a, and at least $\frac{n}{6}$ vertices form windows with base b. These windows form a mesh-like configuration, as shown in the figure. The number of regions is at least $\frac{n^2}{36}$ in the spike. Since there are $\frac{n}{2}$ spikes, there are at least $\frac{n^3}{72}$ visibility regions in the polygon. This construction always give a polygon with an even number of vertices; it is easy to see that the removal of one spike gives a polygon on an odd number of vertices with $\Theta(n^3)$ visibility regions. ∎

The succinct representation of the information contained in the planar subdivision of visibility regions relies on a bound of $O(n^2)$ regions of minimal visibility or sinks. We are able to achieve this by demonstrating more crucial restrictions imposed on the regions by the notion of visibility.

Lemma 3.9 *Given two right windows of point x in polygon P, with r_1 and r_2 representing the two pockets of those windows, no point in r_1 can see a point in r_2.*

Proof: If a point a in r_1 could see a point b in r_2, then the line segment between them must intersect both the window of r_1 and r_2. Consider the window half-planes of r_1 and r_2. Since the two half-planes intersect at x and both windows are right

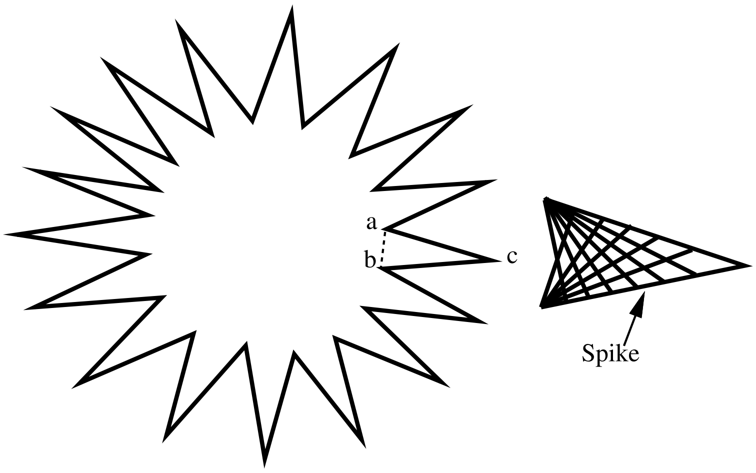

Figure 3.1: Polygon with $\Theta(n^3)$ visibility regions

windows, one of the half-planes must contain the window of the other. The lemma follows from Lemma 2.1. ■

Lemma 3.10 *Given a point x of polygon P. Let $Wind(b, e)$ be a window with respect to point x. If the ray r extending from e away from x and collinear with the window does not intersect $BND(P)$, then let z be any point on r; otherwise let z be the first such intersection point. Then, the line segment $\overline{ez} \in EXT(P)$.*

Proof: Suppose $\overline{ez} \in INT(P)$. This would imply that $\overline{ez}$ is a window with respect to vertex b. This implies that e must be a reflex vertex by Lemma 3.1. Thus, the point x is collinear with two reflex vertices b and e. Either the vertex e cannot be the end of $Wind(b, e)$ according to our refined definition of a window, or our initial assumption that no three vertices of the polygon are collinear is violated. ■

We now generalize Lemma 3.7 which states that any line segment intersects at most one right window with respect to a point x. The following lemma summarizes the essential properties that allow us to show that there are only $O(n^2)$ sinks.

Lemma 3.11 *Let R_x be the set of right windows of point x and R_y be the right windows of point y in polygon P. Then, there is at most 1 intersection between the line segments of R_x and the line segments of R_y.*

Proof: If x is contained in a right pocket of point y, then the lemma holds by Lemma 3.9.

Suppose that x is not contained in a right pocket of point y. Let $r_1 = Wind(b_1, e_1)$ and $r_2 = Wind(b_2, e_2)$ be two windows contained in R_y. Without loss of generality, assume that r_2 is contained in the window half-plane of r_1.

We claim that, if x sees a point $p_1 \in Pocket(b_1, e_1)$ and a point $p_2 \in Pocket(b_2, e_2)$, then x sees b_1. It is clear that, if x sees b_1, then a right window of x cannot intersect r_1.

Let j be the point where the line segment $\overline{xp_1}$ intersects $Wind(b_1, e_1)$. We claim that the line segment $\overline{xp_2}$ must intersect the line segment $\overline{yb_1}$ at a point i. Suppose it does not. Then it intersects either r_1 or the ray extending from r_1 at e_1. If it intersects r_1, then it cannot see p_2, by Lemma 2.1. If it intersects the ray, then it cannot see p_2, by Lemma 3.10.

Having established that the intersection point i exists, the lemma follows since triangle(x, i, j) is contained in P. ■

Theorem 3.12 *A decomposition of a polygon P on n vertices into visibility regions contains $O(n^2)$ sinks and there exist polygons on n vertices which have $\Theta(n^2)$ sinks.*

Proof: Let R_i be the set of all right windows with respect to vertex v_i of polygon P and $R = \bigcup_{i=1}^{n} R_i$. Let L_i and L be defined similarly for left windows.

Let us first consider all the sinks that have an edge which is part of an element in R_i. We claim that there are at most $2n$ such sinks. Let s be a sink in polygon P such that one of the edges of s is contained in a window $r \in R_i$. Let a, b, c be three consecutive vertices of the polygon s in clockwise order with $\overline{ab} \in r$. If $\overline{bc}$ is not in the boundary of P, then it either lies in a right window of a vertex outside the window half-plane of r, or in a left window of a vertex inside r's pocket. We will refer to $\overline{bc}$ as a *sink end.*

If v is contained in a right pocket of vertex v_i with window w then by Lemma 3.9, we know that aside from w, v can see no other element of the set R_i. By Lemma 3.7, we know that only one left window of v can intersect w.

If v is not contained in a right pocket of vertex v_i, then by Lemma 3.11, we know there can only be one intersection between the right windows of v and the right windows of v_i.

There can be at most n sink ends on R_i which are part of the polygon boundary and there can be at most n sink ends on R_i which are part of a left/right window with respect to another vertex. Thus, the total number of possibilities for $\overline{bc}$ is $2n$, that is, the number of sinks associated with R_i is no more than $2n$. Similarly, the number of sinks associated with L_i is no more than $2n$. So the number of sinks associated with a vertex v_i is no more than $4n$. Therefore, the number of sinks in the polygon is no more than $4n^2$.

The polygon in Figure 3.2, is a polygon with $\Theta(n^2)$ sinks. The dark squares represent areas which contain at least one sink and there are $\Theta(n^2)$ dark squares. In order to see that each dark square must contain a sink, observe that moving from the square into a neighboring white square always increases the number of visible vertices. ■

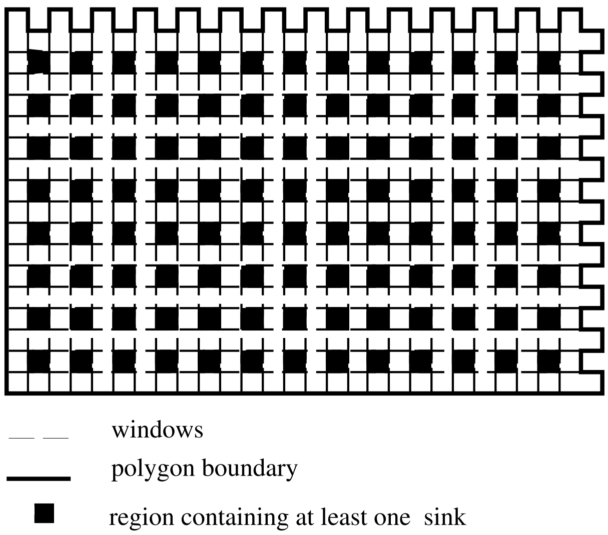

Figure 3.2: Polygon with $\Theta(n^2)$ sinks, one sink in each dark square.

Chapter 4

The Visibility Query Algorithm

In this chapter, we give a high level description of the preprocessing step as described in the overview of Chapter 1.

1. **Construct the set W (as defined in chapter 3).**

   - The set of windows with respect to one vertex in polygon P can be constructed in $O(n)$ time using a standard visibility polygon algorithm (ElGindy and Avis [16], Lee [22], Joe and Simpson [20]). This implies $O(n^2)$ complexity for this step.

2. **Construct the planar subdivision induced by this set.**

   - Although there is an algorithm by Chazelle and Edelsbrunner[11] which finds the planar subdivision induced by a set of line segments in $O(g \log g + i)$ time where g is the number of segments and i is the number of pairwise intersections, the algorithm requires that the line segments be in general position. It is not clear how to guarantee this. However, the much simpler algorithm by Bentley and Ottmann[5] can be easily modified to construct the planar subdivision in time $O(n^3 \log n)$. This slight increase in run time eliminates the need for the line segments to be in general position, but does not affect the overall time complexity since step 3 requires $\Theta(n^3 \log n)$. A detailed analysis of the modifications necessary can be found in [4].

3. **Preprocess the subdivision for planar point location.**

   - Preprocessing for planar point location requires $O(n^3 \log n)$ using standard algorithms (Kirkpatrick [21], Lee and Preparata [24], Preparata [27], and Sarnak and Tarjan [29]).

4. **Construct the dual planar graph of the subdivision.**

   - Constructing the dual planar graph from the planar subdivision can be easily done in $O(n^3)$ time by simply stepping through the planar subdivision in a breadth-first or depth-first manner.

5. **Construct the special dual directed planar graph from the dual planar graph.**

   - Every node in the dual graph represents a visibility region. An edge between two nodes represents a gain of one vertex in the visibility set in one direction and a loss in the other. We convert every edge into a directed edge, where the direction points towards the loss. So a directed arc(a, b) in the directed graph means that region a sees one more vertex than region b. Associate the lost vertex with the arc. This can be done in $O(n^3)$ time.

6. **Associate, with each sink in the special dual directed planar graph, the subset of vertices of P seen.**

   - Notice that a sink in this new directed graph represents a region with minimal visibility set. First, find all the sinks, which requires $O(n^3)$ time. For each sink, associate the subset of vertices of the polygon seen. This can be done by constructing a visibility polygon of any point in the region at a linear cost per sink. Therefore, the cost of this step is $O(n^3)$.

7. **Associate, with each node, the number of vertices seen.**

- This step can be done in $O(n^3)$ by the depth-first traversal of dual graph.

This concludes the high level description of the algorithm. The time complexity is dominated by step 3. The total time complexity of the preprocessing step is $O(n^3 \log n)$. The space required is $O(n^3)$.

Given an arbitrary query point, the algorithm does as follows:

1. locates the region containing the point;
2. starting at the node representing the given region, follows any path leading to a sink.

If we wish to know only the number of vertices seen then only step 1 is necessary as this information is stored in each region. Thus, the time requirement is $O(\log n)$. Now suppose that the length of the path to the sink is s and that there are t vertices in the list associated with the sink. Each arc represents one vertex seen by the query point and the list of vertices of the sink is seen by the query point. Let $k = s + t$. The time required for a query that enumerates the vertices seen is $O(\log n + k)$. The correctness of this approach follows from the discussion in the previous section. Hence, we have

Theorem 4.1 *A simple polygon P can be preprocessed in $O(n^3 \log n)$ time and $O(n^3)$ space such that given an arbitrary query point inside the polygon, it takes $O(\log n + k)$ time to list the k visible vertices and $O(\log n)$ time to give the number of visible vertices.*

Chapter 5

Recovering the Visibility Polygon

5.1 Introduction

Until this point, we have concentrated on the set of vertices seen by a query point. When answering a query, we have ignored the order of the vertices as they appear in the polygon. Recovering the order of the vertices as they appear is an essential step towards recovering the visibility polygon of a query point. To capture this notion of ordering, we need to examine closely some properties of a path in the dual graph from a region that is a sink to a region that is not. These properties will allow us to recover the order of the vertices seen by the query point as they appear in the polygon.

Recall that a visibility region is a maximally connected subset of a polygon with the property that any two points in the region see the same subset of polygon vertices. Two neighboring visibility regions are separated by an edge that is part of a window of some polygon vertex. This common edge belongs to the region which has that polygon vertex in its visibility set.

Let us consider two neighboring visibility regions r_1 and r_2. By Lemma 3.4, we know that the visibility sets of r_1 and r_2 differ by only one vertex. Let that vertex, denoted by v_1, be in the visibility set of r_1. Thus, v_1 is the origin of the window separating the two regions.

The problem we wish to solve is the following. Given the ordered visibility set of r_2, recover the order in the visibility set of r_1. Since the only difference between the sets is the vertex v_1, we need to determine where to insert v_1, in the ordered visibility set of r_2, to recover the ordered visibility set of r_1. We will show that the position of the vertex is directly related to the position of the base of its window. We now give a high-level sketch of the insertion algorithm. Recall that every window in our structure is labelled with the vertex forming the window.

By following a path from the sink to the region containing the query point, we can recover the ordering by following these rules:

1. If a left window is crossed, insert the vertex, in the label of the window, before the base.
2. If a right window is crossed, insert the vertex, in the label of the window, after the base.

We prove the validity of this algorithm in the next section. The order of the vertices is preserved in the sink, so these insertions return the order of the vertices as seen by the query point. Although the algorithm is simple, some care must be taken in its implementation in order to remain within the desired query time complexity.

To recover the visibility polygon of a given query point, we must extend this notion of ordering to include certain key edges. We define the *visibility sequence* of a point x in a polygon P, written as $VS(x, P)$, to be the clockwise ordering of the edges and vertices of $BND(VP(x, P)) \cap BND(P)$. Note that an edge in this context refers to the label of the edge in P and not the exact piece of it that is seen from the point x.

We now refine the definition slightly to easily cope with the ambiguity which arises due to the situation shown in Figure 5.1. If $Wind(s,t)$ is an edge of $VP(x,P)$ and t is a vertex of P, then we insert between s and t in $VS(x,P)$, the edge adjacent to t in P that is in the pocket of $Wind(s,t)$. We are able to extend the insertion algorithm to recover the visibility sequence of a query point. Given the visibility sequence of a query point, we can construct its visibility polygon in linear time in the size of the sequence. Thus, the recovery of the visibility polygon of a query point is within the desired time and space complexity.

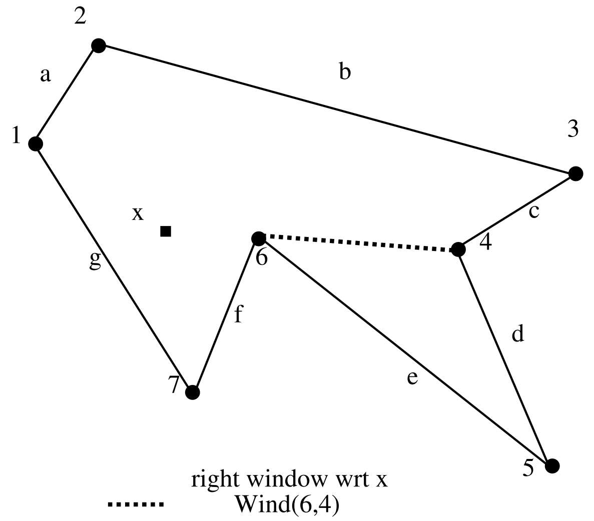

Figure 5.1: $VS(x,P)$ is 1,a,2,b,3,c,4,d,6,f,7,g

5.2 Recovering the Visibility Sequence

In this section, we prove some properties of a path from a sink to an arbitrary region which allows us to recover both the ordered set of vertices and the visibility sequence of the query point. Figure 5.2 serves as an aid to illustrate the lemmas to follow.

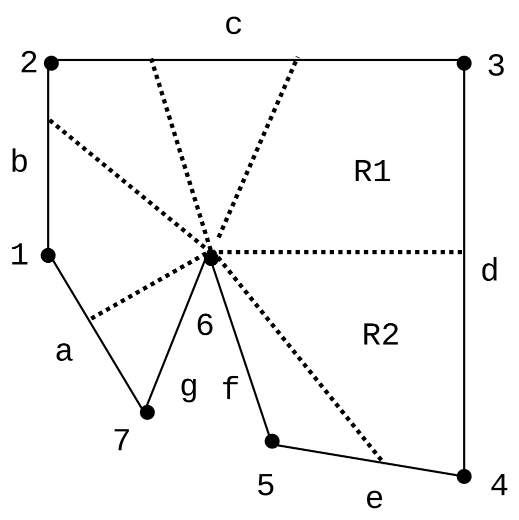

Figure 5.2: A polygon, decomposed into visibility regions, illustrating the lemmas.

Lemma 5.1 *[22] The clockwise (counter-clockwise) order of the vertices of P around the visibility polygon of a point inside a polygon P preserves the clockwise (counter-clockwise) order of the vertices around P.*

We proceed by proving the validity of the simple algorithm described in Section 5.1. Consider two neighboring visibility regions r_1 and r_2 with r_1 being the region containing the common edge. Since the only difference between the two regions is one vertex, say v_1, we need to determine where to insert v_1 in the ordered visibility set of r_2 to recover the ordered visibility set of r_1. The following two lemmas show where the vertex must be inserted.

Lemma 5.2 *Let r_1 and r_2 be two visibility regions separated by a right window. Let r_1 be the region which sees more. Let the right window be a window of vertex v and denote it by $rw(b, e)$ where b is the base and e is the end. The clockwise ordering of the vertices in $VS(r_2, P)$ starting at vertex b with the addition of vertex v*

immediately after b is the same as the clockwise ordering of the vertices in $VS(r_1, P)$ starting at b.

Proof: Consider a point $p \in r_1 \cap rw(b, e)$. Since p is on $rw(b, e)$, the left window $Wind(b, v) \in VP(p, P)$. ■

The case for left windows is symmetric and we have:

Lemma 5.3 *Let r_1 and r_2 be two visibility regions separated by a left window. Let r_1 be the region which sees more. Let the left window be a window of vertex v and denote it by $lw(b, e)$ where b is the base and e is the end. Let v_k be a vertex other than v in the visibility set of r_1 and r_2. The clockwise ordering of the vertices in $VS(r_2, P)$ starting at vertex v_k with the addition of vertex v immediately before b is the same as the clockwise ordering of the vertices in $VS(r_1, P)$ starting at v_k.*

We will now show some properties of a path from a sink to an arbitrary region which will allow us to recover the visibility sequence of a query point. This is the next step towards the recovery of the visibility polygon of a query point.

Lemma 5.4 *Between two consecutive vertices of $VS(x, P)$, there is exactly one edge.*

Proof: Let v_1 and v_2 be the two consecutive vertices of $VS(x, P)$. Let v_k be some other vertex in $VS(x, P)$. Assume without loss of generality that v_1 is followed by v_2 in the clockwise order of the vertices of P starting at v_k. If v_1 and v_2 are consecutive vertices of the polygon P, the lemma holds. So let us assume that v_1 and v_2 are not consecutive polygon vertices. This immediately implies that either v_1 is the base of a window of x or v_2 is the base of a window of x or both. We show that in all cases the lemma holds.

Case 1. Let v_1 be the base of a window of x. Let $w_1(v_1, p_1)$ be that window.

Case 1.1 If $v_2 = p_1$, the lemma holds by the refined definition of $VS(x, P)$.

Case 1.2 Suppose the edge, $e_1 \in BND(VP(x, P))$ preceding v_2, is contained in $VS(x, P)$. If $p_1 \in e_1$, then the lemma holds. So let us assume that $p_1 \notin e_1$. Let p_2 and p_3 be the vertices following p_1 in $VP(x, P)$. Now, p_1 cannot be a vertex of P. The segment $\overline{p_1 p_2}$ must be contained in $BND(P)$. But, p_2 cannot be a vertex of P, therefore, it must be the end of some window. But this implies that p_3 is the base, leading to a contradiction.

Case 1.3 Suppose that the edge, $w_2 \in BND(VP(x, P))$ preceding v_2, is not contained in $VS(x, P)$; then w_2 is a window with base v_2. Let p_2 be the end of w_2. If $\overline{p_1 p_2} \in BND(P)$, then the lemma holds. So let us assume that $\overline{p_1 p_2} \notin BND(P)$. But this implies that $CWC(p_1, p_2) \in BND(VP(x, P))$ must contain a vertex of P, leading to a contradiction.

All other cases are symmetric. ∎

Although, by Lemma 5.1, we see that the vertex ordering of any two points in a visibility region is the same, it is not clear whether the visibility sequence is also the same. We will now show that, if two points are in the same visibility region, then they have the same visibility sequence.

Lemma 5.5 *Two points in the same visibility region have the same visibility sequence.*

Proof: Let x_1 and x_2 be two points in a polygon P. Let v_1 and v_2 be two consecutive vertices in $VS(x_1, P)$ and in $VS(x_2, P)$. We need to show that if the edge between v_1 and v_2 in $VS(x_1, P)$ is different from the edge between them in $VS(x_2, P)$, then x_1 and x_2 are not in the same visibility region. We do this by showing that either x_1 does not see x_2 or there is a window with base v_1 or v_2 which separates x_1 from x_2.

Let us assume that x_1 sees x_2. Thus, we only need to prove that the difference in edges implies that there is a window with base v_1 or v_2 which separates x_1 from x_2. The two vertices v_1 and v_2 cannot be consecutive in P, since this would imply that the edge between them is the same for x_1 and x_2 by Lemma 3.6.

Let e_1 and e_2 be the two different edges between v_1 and v_2 in the visibility sequences of x_1 and x_2 respectively. Since the vertices v_1 and v_2 are not consecutive polygon vertices, either one or both vertices must be bases of windows in $VP(x_1, P)$ and $VP(x_2, P)$. We now show that in all cases there is a window with base v_1 or v_2 which separates the two points x_1 and x_2.

Case 1. The vertex v_1 is a base of a window for both x_1 and x_2. Let the window for x_1 be $w_1(v_1, d_1)$ and the window for x_2 be $w_2(v_1, d_2)$. The end d_1 is contained in e_1 and d_2 is contained in e_2. If w_1 is a left window and w_2 is a right window, then both the primary left window with base v_1 and the primary right window with base v_1 separate x_1 from x_2.

Let us suppose that w_1 and w_2 are of the same type. Assume, without loss of generality, that e_1 comes before e_2 in the clockwise order starting from v_1 in P. The $CWC(d_1, d_2)$ of the $VP(v_1, P)$ must contain a vertex of P by lemma 5.4. Let y be the first such vertex. The line defined by the line segment $\overline{yv_1}$ intersects the line segment $\overline{x_1x_2}$. Thus, there is a window of y with base v_1 separating x_1 from x_2.

Case 2. The vertex v_1 is not the base of a window for x_1 but it is for x_2. Let the window for x_2 be $w_2(v_1, d_2)$. Since v_1 is not a base for x_1, v_1 must be an endpoint of the edge e_1 in the polygon P.

Again, assume, without loss of generality, that e_1 comes before e_2 in the clockwise order of P starting at v_1. The $CWC(v_1, d_2)$ of the $VP(v_1, P)$ must contain a vertex of P by Lemma 5.4. Let y be the first such vertex. The line defined by the line segment $\overline{yv_1}$ intersects the line segment $\overline{x_1x_2}$. Thus, there is a window of y with base v_1 separating x_1 from x_2.

All other cases are symmetric. ∎

We will refer to the visibility sequence of a visibility region r as $VS(r, P)$.

Lemma 5.6 *Let r_1 and r_2 be two visibility regions separated by a right window. Let r_1 be the region which sees more. Let the right window be a window of vertex v and denote it by $rw(b, e)$ where b is the base and e is the end. Let d be the edge just before v in clockwise ordering of $VS(b, P)$ starting at vertex b. The clockwise ordering of $VS(r_2, P)$ starting at vertex b with the addition of dv immediately after b is the same as the clockwise ordering of $VS(r_1, P)$ starting at b.*

Proof: Let $S_{r_1} = VS(r_1, P) - \{d, v\}$. Suppose there is a difference between S_{r_1} and $VS(r_2, P)$. The difference must be an edge difference since the only vertex difference was removed. Let v_1 and v_2 be the pair of consecutive vertices which harbor the edge difference between them. By Lemma 5.5, there is a window with base v_1 or v_2 separating r_1 and r_2. But this implies that there are two windows separating r_1 and r_2 which contradicts the fact that they are adjacent regions. Therefore, $S_{r_1} = VS(r_1, P)$.

By Lemma 5.2, we know that v occurs immediately after b in the clockwise order starting at b. Since d is the edge between b and v in $VS(r_1, P)$, the lemma holds. ■

Again, the case for left windows is symmetric and we have:

Lemma 5.7 *Let r_1 and r_2 be two visibility regions separated by a left window. Let r_1 be the region which sees more. Let the left window be a window of vertex v and denote it by $lw(b, e)$ where b is the base and e is the end. Let v_k be a vertex other than v in the visibility set of r_1 and r_2. Let d be the edge just after v in clockwise ordering of $VS(b, P)$ starting at vertex b. The clockwise ordering of $VS(r_2, P)$ starting at vertex v_k with the addition of vd immediately before b is the same as the clockwise ordering of $VS(r_1, P)$ starting at v_k.*

We will describe the algorithm to recover the visibility sequence of a query point in time $O(\log n + k)$, where k is the size of the sequence. First, we must enhance our digraph structure. Between two neighboring visibility regions, the common edge between them is represented by an arc in the digraph and the arc is labelled with the vertex that represents the sole difference between the visibility sets. Since we want to recover the sequence, we must label the arc with the sole polygon vertex as well as the sole polygon edge representing the difference between the two visibility regions.

The labelling will be slightly different depending on the type of window. Let $rw(b, e)$ be a right window of vertex v, with b as the base and e as the end of the window. Let d be the edge just before v in the clockwise ordering of $VS(b, P)$ starting at vertex b. The window will have label dv. Let $lw(b, e)$ be a left window of vertex v, with b as the base and e as the end of the window. Let d be the edge just after v in the clockwise ordering of $VS(b, P)$ starting at vertex b. The window will have label vd.

By Lemma 5.6 and Lemma 5.7, we see that by following a path from the sink to the region containing the query point, we can recover the visibility sequence by following some simple rules. Notice that the rules are a slight modification of the rules to recover the ordered set of vertices.

1. If a left window is crossed, insert the vertex followed by the edge, in the label of the window, before the base.
2. If a right window is crossed, insert the edge followed by the vertex, in the label of the window after the base.

Recall that a polygon P on n vertices is denoted by a set of vertices $v_1, v_2, \ldots, v_{n-1}, v_n$ such that every two consecutive vertices are joined by an edge, including the pair $\{v_n, v_1\}$. For ease of explanation, we will refer to the vertices by their indices. We will now sketch the implementation.

Let us start by concentrating on the structure stored in a sink. When we were solely concerned with the recovery of the set of vertices visible from a query point, we simply stored the set of vertices visible from the sink in an array. To recover the visibility sequence from a query point, the structure of the set stored in a sink must be enhanced in the following way.

The visibility sequence of the sink is stored in order in a doubly linked list with a special node pointing to the head of the list. An array of $2n$ pointers, indexed by the indices of the vertices and edges ordered as they appear in the polygon, points to the appropriate nodes in the doubly linked list. If the node is not in the linked list, then the pointer in the array will be a null pointer. For example, figure 5.3 shows the structure stored in sink $r1$.

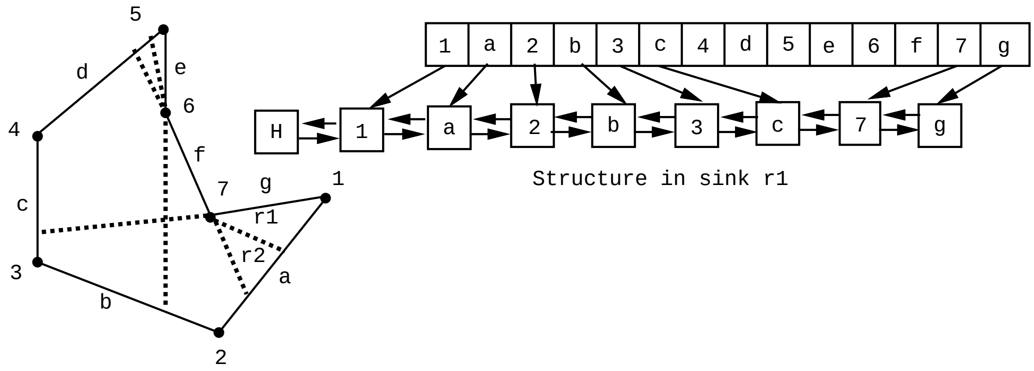

Figure 5.3: Polygon decomposed into visibility regions and enhanced structure stored in sink.

The problem now becomes to step from the sink back to the region containing the query point in the digraph structure, while updating the doubly linked list. When the region containing the query point is reached, the doubly linked list will have the visibility sequence as seen by the query point.

Let us look at one step as taken in figure 5.3. The first step from region 1 to region 2 crosses a left window of vertex 4 with base 7. Since the window is a left window, its label will be $4d$. By the simple rules derived from Lemma 5.7, we know that $4d$ must be inserted before 7. The insertion is quite straightforward. We first find node 7 in the doubly linked list by referring to the array of pointers with index 7.

Node 4 and node d is then inserted into doubly linked list and a pointer to node 4 and node d is added to array of pointers with index 4 and d respectively. See Figure 5.4 to see what the structure looks like after one update. Thus, one step requires $O(1)$ time.

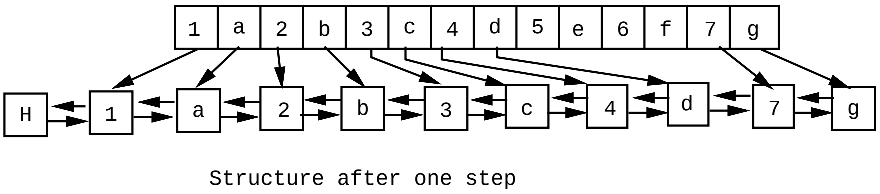

Figure 5.4: Structure after one update.

Hence, we have:

Theorem 5.8 *A simple polygon P can be preprocessed in $O(n^3 \log n)$ time and $O(n^3)$ space such that given an arbitrary query point inside the polygon, $O(\log n + k)$ time is used to recover the visibility sequence of size k.*

Given the visibility sequence of a query point q, recovering the actual visibility polygon requires that we compute the vertices of $VP(q, P)$ which are not vertices of P, the edges of $VP(q, P)$ which are not edges of P, and the pieces of edges of P seen by q. By Lemma 5.4, we know that there is exactly one edge between two consecutive vertices of the visibility sequence. Let v_1 and v_2 be the two consecutive vertices and let e_1 be the edge between them. If the two vertices are consecutive polyon vertices, then no computation is needed, because the edge between them is the edge in the visibility polygon. If the two vertices are not consecutive polygon vertices, then the intersection, i_1, of the line formed by $\overline{v_1 q}$ and edge e_1 as well as the intersection, i_2, of the line formed by $\overline{v_2 q}$ and e_1 must be computed, because the portion of the visibility polygon of query point q is denoted by v_1, i_1, i_2, v_2. Thus, we have:

Theorem 5.9 *A simple polygon P can be preprocessed in $O(n^3 \log n)$ time and $O(n^3)$ space such that given an arbitrary query point inside the polygon, $O(\log n + k)$ time is required to recover the visibility polygon of size k.*

Chapter 6

Extensions

6.1 Query points outside the polygon

In this section, we refine the solution to handle query points in the *exterior* as well as the interior of the polyon. To avoid the ambiguity between exterior and interior visibility, a point on the boundary of the polygon will be considered to be in the interior. If a given query point is inside the polygon, we answer the query using the solution already developed. We now turn our attention to query points which are strictly outside a polygon and satisfy the queries with respect to exterior visibility.

We define the *convex hull* of a polygon P to be the smallest convex polygon, say $CH(P)$, containing P. A vertex on $CH(P)$ is known as a *convex hull vertex*. If an edge of $CH(P)$ is not an edge of P, we say it is a *bay edge*. If $\overline{v_a v_b}$ is a bay edge, then the corresponding *bay* is the polygon formed of the vertices $(v_a, v_b, v_{b-1}, v_{b-2}, \ldots, v_{a+2}, v_{a+1}, v_a)$. A vertex on a bay, but not on the convex hull, is known as a *bay vertex*.

The general idea is to divide the exterior points into two classes: exterior points inside bays and exterior points outside the convex hull of the polygon. Since a bay is a polygon, it can be preprocessed using the techniques developed; this takes care of query points inside bays. The following results show that the time and space complexities necessary to preprocess the bays are $O(n^3 \log n)$ and $O(n^3)$, respectively.

Observation 6.1 *A bay vertex can be in only one bay.*

Observation 6.2 *A convex hull vertex can be in at most two bays.*

Lemma 6.3 *The time required to preprocess all of the bays is $O(n^3 \log n)$ and the space required is $O(n^3)$.*

Proof: Let β be the number of bays in a polygon P on n vertices. Let b_i be the number of vertices in the i^{th} bay including the convex hull vertices. By Observation 6.1 and Observation 6.2, we deduce that

$$\sum_{i=1}^{\beta} b_i \leq 2n,$$

but this implies that

$$\sum_{i=1}^{\beta} b_i^3 \log b_i \leq 2n^3 \log n.$$

By the same argument, the total space required to store all the information is $O(n^3)$. The lemma follows. ∎

Therefore, by preprocessing each bay using the techniques developed so far, we are still within the desired overall time and space complexity. Turning our attention to the structure outside the convex hull, we obtain the following results:

Lemma 6.4 *A convex-hull vertex of a polygon P on n vertices can have at most two of its windows intersecting the outside of the convex hull.*

Proof: The convex hull vertex, v, can be considered as a vertex of a convex polygon. When viewed from the exterior, a convex hull vertex is actually a reflex vertex. By Lemma 3.1, the base of any window must be a reflex vertex. Since there are no reflex vertices outside the convex hull, the only windows of the hull vertex are those emanating from the two hull vertices adjacent to v. ■

Lemma 6.5 *A bay vertex of a polygon P on n vertices can have at most two of its windows intersecting the outside of the convex hull.*

Proof: By Lemma 3.1, the base of any window must be a reflex vertex. Since there are no reflex vertices outside the convex hull, the only windows intersecting the outside of the convex hull are the windows intersecting the bay edge. By Lemma 3.7, there can be at most 2 windows. ■

Since a polygon vertex can only be a bay vertex or a convex hull vertex, these two lemmas imply that there are at most $2n$ windows outside the convex hull. This means that the planar subdivision induced by these windows has only $O(n^2)$ regions. We do not even need to worry about sinks. By simply placing the necessary information in each visibility region outside the convex hull, we remain within the desired time complexity.

We must refine the planar point location, though. We must have a first pass in the planar point location to decide whether the query point is in the polygon, a bay or outside the convex hull. There are only a linear number of bays, so the preprocessing for the first pass in the planar point location will require only $O(n \log n)$ and $O(n)$ space with $O(\log n)$ query time. Once this has been determined, we use the techniques previously described in order to extract the necessary information. The query time is still $O(\log n + k)$ to list the k visible points or give the whole visibility polygon of size k and $O(\log n)$ to recover only the number of visible points. Thus, we have:

Theorem 6.6 *A simple polygon P can be preprocessed in $O(n^3 \log n)$ time and*

$O(n^3)$ space such that given an arbitrary query point inside or outside the polygon, $O(\log n + k)$ time is required to list the k visible vertices or recover the visibility polygon of size k and $O(\log n)$ time to give the number of visible vertices.

6.2 Visibility of Fixed Points

In this section, we develop a technique to deal with visibility queries concerning a fixed set of points. The techniques used previously decompose the polygon with respect to polygon vertex visibility. We will show how to generalize this to decompose a polygon with respect to a set of fixed points in the plane rather than the polygon vertices. We will concentrate only on a fixed set of points contained within the polygon, since the extension to fixed points both inside and outside the polygon follows from the discussion in section 6.1. Intuitively, we can think of the set of fixed points as a set of point light sources. In the query, we are simply asking for the light sources which are shining light on the query point. For simplicity of presentation, we will assume that no three points are collinear among the fixed points and polygon vertices.

We must generalize the notion of a visibility region to the following: Given a set S of fixed points contained in a simple polygon P, a *visibility region* R, in P is a *maximally connected* subset of P with the property that any two points in R see the same subset of fixed points from S. All other terms are defined similarly.

Let $\|S\| = m$. Let the points in S be labelled $s_1 \ldots s_m$. Analogous to the definition of the set W in chapter 3, we define a set W'. Let W'_i be the set of all windows in P with respect to point s_i. Let

$$W' = BND(P) \cup \bigcup_{i=1}^{n} W'_i$$

Consider the planar subdivision induced by all of the line segments in W'. The lemmas that follow are generalizations of the lemmas presented in Chapter 3.

Lemma 6.7 *The set of non-external faces in such a subdivision is precisely the set of visibility regions.*

Proof: First, we show that an arbitrary non-external face f in such a subdivision is contained in a visibility region, r, of the polygon. We know that a face in a planar subdivision is a connected region. Thus, if f was not contained in a visibility region, then there must exist two points in f that do not see the same subset of points of S. Let x and y be two such points. Without loss of generality, let v be a point of the set S which is seen by x but not y. This implies that y lies in a hidden region with respect to v. Therefore, since y is in a hidden region with respect to v and x is not, they cannot lie on the same face.

Now we show that an arbitrary visibility region, r, of the polygon is contained in a non-external face, f. Suppose that a visibility region r was not a non-external face of such a subdivision. There must exist two points in r that lie in two adjacent non-external faces. Let x and y be two such points, lying in faces f_1 and f_2 respectively. The faces f_1 and f_2 are separated by an edge which is part of some window. Let that window be a window with respect to point v of S. But this implies that one of x or y does not see v. ■

Corollary 6.8 *Given two visibility regions that share a common edge, their visibility set is the same set except for one point from the set S.*

Theorem 6.9 *With respect to a set of fixed points $\|S\| = m$, a decomposition of a polygon P on n vertices into visibility regions contains $O(m^2n)$ regions, and there exist some polygons which have $\Theta(m^2n)$ visibility regions.*

Proof: By lemma 3.2, we have

$$\|W'\| \leq mn$$

Let i be the number of intersections on a single window from the set W'. By lemma 3.6, we infer that $i < 2m$. Therefore, the total number of intersections in a decomposition is less than $2m^2n$. Euler's Formula ([6], p. 143) states that if G is a connected plane graph, then $v - e + f = 2$, where v is the number of vertices, e is the number of edges, and f is the number of faces of G. Hence, $e \leq 3v - 6$ and we deduce that there are no more than $4m^2n$ visibility regions.

The polygon in figure 6.1, is a polygon with $\Theta(m^2n)$ visibility regions. There are $O(m/n)$ fixed points in each spike, represented by the black dots. This causes $\Theta(m^2)$ regions per spike and there are $O(n)$ spikes. ■

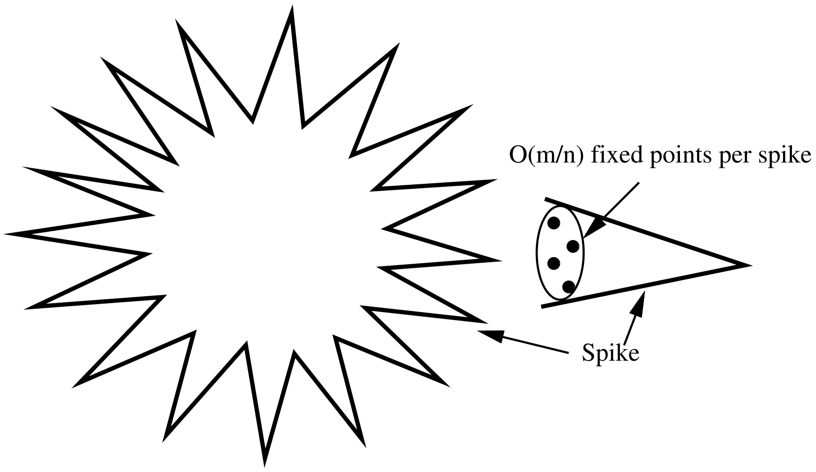

Figure 6.1: Polygon containing $\Theta(m^2n)$ visibility regions.

Theorem 6.10 *With respect to a set of fixed points $\|S\| = m$, a decomposition of a polygon P on n vertices into visibility regions contains $O(m(m+n))$ sinks.*

Proof: Let R_i be the set of all right windows with respect to point s_i of set S. Let $R = \bigcup_{i=1}^{n} R_i$. Let L_i and L be defined similarly for left windows.

Let us first consider all of the sinks associated with R_i. We claim that there are at most $m + n$ sinks which have an edge that is part of an element of R_i. Let sk be a sink in polygon P such that one of the edges of sk is contained in a window $r \in R_i$.

Let a, b, c be three consecutive vertices of the polygon sk in clockwise order with $\overline{ab} \in r$. If $\overline{bc}$ is not on the boundary of P, then it either lies in a right window of a point of S outside the window half-plane of r, or in a left window of a point of S inside r's pocket. We will refer to $\overline{bc}$ as a *sink end.*

If a point s is contained in a right pocket of point s_i with window w, then by Lemma 3.9, we know that aside from w, s can see no other element of the set R_i. By lemma 3.7, we know that only one left window of s can intersect w.

If s is not contained in a right pocket of point s_i, then by Lemma 3.11, we know there can only be one intersection between the right windows of s and the right windows of s_i.

There can be at most n sink ends on R_i which are part of the polygon boundary since there can only be at most n elements in R_i. There can be at most m sink ends on R_i which are part of a left/right window with respect to another point of S because S has m elements. Thus, the total number of possibilities for $\overline{bc}$ is $m + n$, that is, the number of sinks associated with R_i is no more than $m + n$. Similarly, the number of sinks associated with L_i is no more than $m + n$. So the number of sinks associated with a point s_i is no more than $2(m + n)$. Therefore, the number of sinks in the polygon is no more than $2m(m + n)$. ■

Thus, we have:

Theorem 6.11 *Let $\|S\| = m$ be a set fixed points in the plane and P be a simple polygon with n vertices. Given an arbitrary query point inside or outside the polygon, we can recover the number of visible fixed points in $O(\log mn)$ time and recover the visible set of k fixed points in $O(\log n + k)$ time, with $O(m^2(m+n)\log n)$ preprocessing time and $O(m^2(m + n))$ space.*

6.3 Line Segment Queries

Now, we revert to visibility queries concerning polygon vertices for simplicity of presentation, but instead of query points, we have line segment queries. When dealing with line segments, there exist two notions of visibility. We might want to know the set of polygon vertices which see the whole line segment, known as *strong visibility*, or we might want to know the set of polygon vertices which see any part of the line segment, known as *weak visibility*. We consider the problem of weak visibility; strong visibility of line segments seems to be a more difficult problem.

To recover the weak visibility set, we determine the set of vertices visible to one end point. Let us call this set WVS. The line segment will intersect a set of regions in the decomposition. We follow the path formed by regions which are intersected by the line segment. If a window is crossed and visibility is gained, then the vertex is added to the WVS. If a window is crossed and visibility is lost, then nothing is done. When the region containing the other end point is reached, the set WVS represents the set of vertices weakly visible from the line segment.

The only difficulty is finding the path formed by the line segment. Since each visibility region is convex, the problem reduces to the following: Given a point inside a convex region and a direction, find which edge of the region is intersected. This intersection is easily found by performing a binary search on the ordered set of edges of the convex region. Thus, we have

Theorem 6.12 *A simple polygon P can be preprocessed in $O(n^3 \log n)$ time and $O(n^3)$ space such that given an arbitrary query line segment inside the polygon, $O(k \log n)$ time is required to recover k weakly visible vertices.*

The extension to visibility of a fixed set of points follows from discussion in Section 6.2. The extension to line segment queries outside the polygon follows from the discussion in Section 6.1.

6.4 An Alternative Approach Using Persistence

We present a method using persistent data structures that gives an alternative solution to all of the problems we have been considering so far. The solutions are similar except that the persistent data structure replaces the special directed dual planar graph. This removes some of the geometric intuition gained from the knowledge of sink regions.

By Lemma 3.7, we know that each window in the planar subdivision has at most $O(n)$ intersections with other windows, where n is the number of vertices in the polygon P. Let us focus our attention on a single window, say W, in this planar subdivision. Between the two endpoints of a window are the intersection points with other windows. Let $p_1 \ldots p_r$ be the consecutive ordering of these points of interest with p_1 being one endpoint and p_r the other. Aside from the endpoints, all other points represent an intersection between the window W and some other window. A window forms a boundary between a region seen by a vertex and a region not seen by a vertex. From this, we deduce that each $\overline{p_i p_{i+1}}$ where $1 < i < r-1$ forms the boundary of a different visibility region, each adjacent to the next.

Let L be the set of vertices seen from p_1 on window W. Since the point p_2 represents an intersection with another window, the set of vertices seen from any point between p_2 and p_3 is L with the addition or deletion of one vertex, by Lemma 3.4. Thus, each intersection point identifies an update to the list L. We will sketch the use of a persistent data structure.

If we consider updates to a data structure as steps in time, then a persistent data structure allows access to any version of the data structure at any time. A structure is partially persistent if one can only update the structure in the present (queries are allowed in the past and present). A structure is fully persistent if we can update the structure both in the past and the present. In this situation, we need only a partially persistent data structure to handle queries and updates of insertions, deletions. In particular, Sarnak and Tarjan [29] develop persistent search trees, which have the property that $O(n)$ updates to an empty persistent search tree can be recorded in $O(n \log n)$ time and $O(n)$ space. Therefore, the information for each window can be stored in $O(n \log n)$ time and $O(n)$ space. There are $O(n^2)$ windows which implies $O(n^3)$ total storage and $O(n^3 \log n)$ time.

In order to recover the vertices seen by the query point, a planar point location query will identify the region containing the point. Given the region, choose one of the windows forming the region boundary. To recover the visible vertices, we perform an inorder traversal of the tree associated with that window for that time represented by a point of the region on the window. The extension to visibility sequence queries, visibility polygon queries, fixed point queries, and visibility outside a polygon follows naturally from the discussions developing the solutions using the dual planar graph.

Chapter 7

Conclusion

In this thesis, we have presented a new method of decomposition which allows the preprocessing of a simple polygon to efficiently answer queries of various forms. With $O(n^3 \log n)$ preprocessing time and $O(n^3)$ space, we can, given a query point inside or outside the polygon, recover the number of visible vertices in $O(\log n)$ time and recover the visibility polygon of size k in $O(\log n + k)$ time. Given m fixed points in the plane, and a polygon of size n, we can, given a query point inside or outside the polygon, recover the number of fixed points unobstructed by the polygon boundary in $O(\log mn)$ time and recover the visible set of size k in $O(\log mn + k)$, with $O(m^2(m+n) \log n)$ preprocessing time and $O(m^2(m+n))$ space.

We have also looked at line segment queries. Given a line segment, we can recover its weak visibility set and more generally its weak visibility sequence in $O(k \log n)$, with $O(n^3 \log n)$ preprocessing time and $O(n^3)$ space. To answer weak visibility line segment queries on a set of m fixed points in the plane, we need $O(k \log mn)$ query time, given $O(m^2(m+n) \log n)$ preprocessing time and $O(m^2(m+n))$ space.

There are several questions which are generated from this investigation:

1. Can a non-trivial lower bound be proved for this problem?
2. Can we remove the $\log n$ factor in the time complexity? This would mean that both the time and space complexity are optimal with respect to the decomposition.
3. Is there a smooth trade-off between query time and preprocessing time?
4. Can we recover the strong visibility set of a line segment query in a $O(k \log n)$ time using the same amount of preprocessing as in the weak visibility case?

Bibliography

[1] A. Aggarwal. *The art gallery theorem: Its variations, applications, and algorithmic aspects.* Ph.D. thesis, Johns Hopkins University, 1984.

[2] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. *Visibility of Disjoint Polygons.* Algorithmica, **1**, pp. 49-63, 1986.

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

[4] P. Belleville. *Computing Two-Covers of Simple Polygons.* Masters Thesis, McGill University, 1991.

[5] J.L. Bentley and Th. Ottmann. *Algorithms for reporting and counting geometric intersections.* IEEE Transactions on Computers, **28**, pp. 643-647, 1979.

[6] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications.* Elsevier Science, New York, New York, 1976.

[7] B. Chazelle. *A Theorem on Polygon Cutting with Applications.* Proceedings of the 23rd Annual IEEE Symposium on the Foundations of Computer Science, pp. 339-349, 1982.

[8] B. Chazelle. *An Improved Algorithm of the Fixed-Radius Neighbor Problem.* Infomation Processing Letters, **16**, pp. 193-198, 1983.

[9] B.Chazelle and L. Guibas. *Visibility and Intersection Problems in Plane Geometry.* Discrete and Computational Geometry, **4**, pp. 551-581, 1989.

[10] B. Chazelle, L. Guibas, and D. Lee. *The Power of Geometric Duality.* Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science, pp. 217-225, 1983.

[11] B. Chazelle and H. Edelsbrunner. *An Optimal Algorithm for Intersecting Line Segments in the Plane.* Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, pp. 217-225, 1988.

[12] V. Chvátal. *A combinatorial theorem in plane geometry.* J. Comb. Theory Ser. B, **18**, pp. 39-41, 1975.

[13] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Mass., 1990.

[14] D. Dobkin and R. Lipton. *Multidimensional Searching Problems.* SIAM Journal of Computing, **5, 2**, pp. 181-186, 1976.

[15] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, New York, New York, 1987.

[16] H. ElGindy and D. Avis. *A Linear Algorithm for Computing the Visibility Polygon from a Point.* Journal of Algorithms, **2**, pp. 186-197, 1981.

[17] L. Guibas, J. Hershberger, D. Leven, M. Sharir, R. Tarjan. *Linear Time Algorithms for Visibility and Shortest Path Problems inside Simple Polygons.* Proceedings of the 2nd Annual ACM Symposium on Computational Geometry, pp. 1-12, 1986.

[18] L. Guibas, R. Motwani, and P. Raghavan. *The Robot Localization Problem in Two Dimensions.* To appear in Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms. 1992.

[19] J. Hershberger. *An Optimal Visibility Graph Algorithm of Triangulated Simple Polygons.* Algorithmica, **4**, pp. 141-155, 1989.

[20] B. Joe and R.B. Simpson. *Visibility of a simple polygon from a point.* University of Waterloo technical report, 1985.

[21] D. Kirkpatrick. *Optimal Search in Planar Subdivisions.* SIAM Journal of Computing, **12, 1**, pp. 28-35, 1983.

[22] D.T. Lee. *Visibility of a Simple Polygon.* Computer Vision, Graphics, and Image Processing, **22**, pp. 207-221, 1983.

[23] D.T. Lee and A.K. Lin. *Computing the Visibility Polygon from an Edge.* Computer Vision, Graphics, and Image Processing, **34**, pp. 1-19, 1986.

[24] D.T. Lee and F.P.Preparata. *Location of a Point in a Planar Subdivision and its Applications.* SIAM Journal of Computing, **6, 3**, pp. 594-606, 1977.

[25] K. Mehlhorn. *Multi-dimensional Searching and Computational Geometry.* Springer-Verlag, New York, New York, 1984.

[26] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, New York, 1987.

[27] F.P.Preparata. *A New Approach to Planar Point Location.* SIAM Journal of Computing, **10, 3**, pp. 473-482, 1981.

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

[29] N. Sarnak and R. Tarjan. *Planar Point Location Using Persistent Search Trees.* Communications of the ACM, **29, 7**, pp. 669-679, 1986.

[30] T.C. Shermer. *Visibility Properties of Polygons.* Ph.D. Thesis, McGill University, 1989.

[31] G. Toussaint. *Computational Geometry.* Elsevier Science, Amsterdam, Netherlands, 1985.