Efficient Visibility Queries in Simple Polygons*

Prosenjit Bose‡ Anna Lubiw§ J. Ian Munro§

Abstract

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

The key notion behind the decomposition is the succinct representation of *visibility regions*, and tight bounds on the number of such regions. These techniques are extended to handle other types of queries, such as visibility of fixed points other than the polygon vertices, and for visibility from a line segment rather than a point. Some of these results have been obtained independently by Guibas, Motwani and Raghavan [18].

1 Introduction

The notion of visibility among geometric objects is a fundamental topic in computational geometry. Many different visibility problems have been studied in the literature. The ray shooting problem (given a polygon and a ray emanating from a given point in a given direction, find the first intersection, if any, of the ray with the polygon boundary) has been studied by Chazelle [10], Chazelle and Guibas [12], Guibas et al [17], and Hershberger and Suri [20]. The guarding problem (given a polygon, find a placement of guards at the vertices to cover the whole polygon) and its many variations has been studied by Chvátal [15], Aggarwal [1], O'Rourke [27], and Shermer [31]. A survey paper by Shermer [32] reviews the recent results in this area. The visibility polygon problem (given a polygon P and a point q inside the polygon, determine the polygon consisting of all the points in P visible from q) has been studied by ElGindy and Avis [16], Lee [23], Joe and Simpson [21]. O'Rourke's book [27] surveys many visibility problems which have been investigated.

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 single-shot version of this problem is a special case of the visibility polygon problem and can be solved in $O(n)$ time where n is the number of vertices in the polygon. In this paper, we develop a data structure that can be used to answer repeated queries efficiently. After the data structure

*Research supported in part by NSERC and ITRC. Preliminary versions of this work appear in [6, 7]

‡School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada, K1S 5B6. E-mail address: jit@scs.carleton.ca

§Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1. E-mail address: alubiw@watmath.uwaterloo.ca, imunro@watdragon.uwaterloo.ca

has been built, the query algorithm can find the number of vertices of P which see q in $O(\log n)$ time. 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*. We show that there can be $\Theta(n^3)$ distinct visibility regions, each of which can see $\Theta(n)$ points. The main theorem is that there can be $\Theta(n^2)$ regions with *minimal visibility sets*. 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. Our structure is related to a well-studied structure in 3-dimensions known as the *aspect graph*, which is a graph that represents the partitioning of 3-space into maximal regions so that the views from all viewpoints in the region have the same *aspect*, where the last concept is application dependent, but could for example be the same combinatorial type, (see [9] for a survey).

We uncover some additional properties of the decomposition that allow us to recover the visibility polygon of the query point as opposed to just the visible vertices using virtually the same data structure. The visibility polygon represents all the points in the polygon that see the query point. Our main result states that with $O(n^3 \log n)$ preprocessing time and $O(n^3)$ space, we can recover the visibility polygon of a query point in $O(\log n + k)$ time, where k is the size of the visibility polygon.

Independently of our work [6, 7], Guibas et al. [18] developed the notion of decomposing a polygon into visibility regions while solving the Robot Localization Problem which in some sense is the inverse of the problem considered above. Whereas we consider the problem of recovering the visibility polygon of a given query point, they consider the problem of locating a point inside a polygon given that point's visibility polygon. They solve this problem using a decomposition that is similar to the visibility decomposition we propose, thus some overlap occurs in uncovering the structural properties of the visibility decomposition given in Section 3. They note that their decomposition can be used to solve the visibility polygon query problem.

We can modify the data structure to handle queries with respect to a set of fixed points. Given a fixed set of m points S in the plane, an n vertex polygon polygon, and a query point, we are interested in the subset of S seen from the query point unobstructed by the boundary of the polygon. 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 are able to extend the structure, without additional preprocessing time, to deal with line segment queries. In particular we look at weak visibility queries with respect to line segments. A point is weakly visible from a line segment if it sees any part of the line segment. Thus, given a line segment inside or outside a polygon of size n, we can report all vertices that are weakly in $O(k \log n)$ time. 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.

1.1 Overview

We first give an informal overview of the approach used. 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 R is the subset of vertices of P seen from R. Two visibility regions are *neighboring* if they are separated by an 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 become clear.

A simple argument shows that the decomposition of a polygon into visibility regions has size $O(n^4)$. A more refined counting argument, based on the simplicity of P, shows that there are in fact $O(n^3)$ visibility regions, and this is tight in the worst case. This immediately offers a solution to the query problem. Simply associate with each region the subset of vertices it sees. When given a query point, 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 (see Figure 1). 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 with 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 labeling the edges of π. This notion was first used by Chazelle [11] in a different context.

Another technical counting argument allows us to establish that there are $O(n^2)$ sinks or regions with a minimal visibility set and the bound is tight in the worst case. 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.

The general approach is outlined as follows: During the preprocessing phase, decompose the polygon into visibility regions. Preprocess the regions for planar point location (Kirkpatrick [22], Lee and Preparata [25], Preparata [28], and Sarnak and Tarjan [30]). Build the directed dual planar graph of the planar subdivision. For each sink, store the subset of vertices of P seen by that sink. At query time, a standard $O(\log n)$ planar point location query reveals the region containing the query point. Once the region is identified, we enumerate the vertices by following a path from the region's node to any sink. The preprocessing time is $O(n^3 \log n)$ since the cost is dominated by the time used 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.

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 [27], Bondy and Murty [8], Preparata and Shamos [29]. 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 terminology used in this paper.

2.1 Geometric Terminology

Let E^2 denote the Euclidean space of dimension 2. 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 paper. We will denote

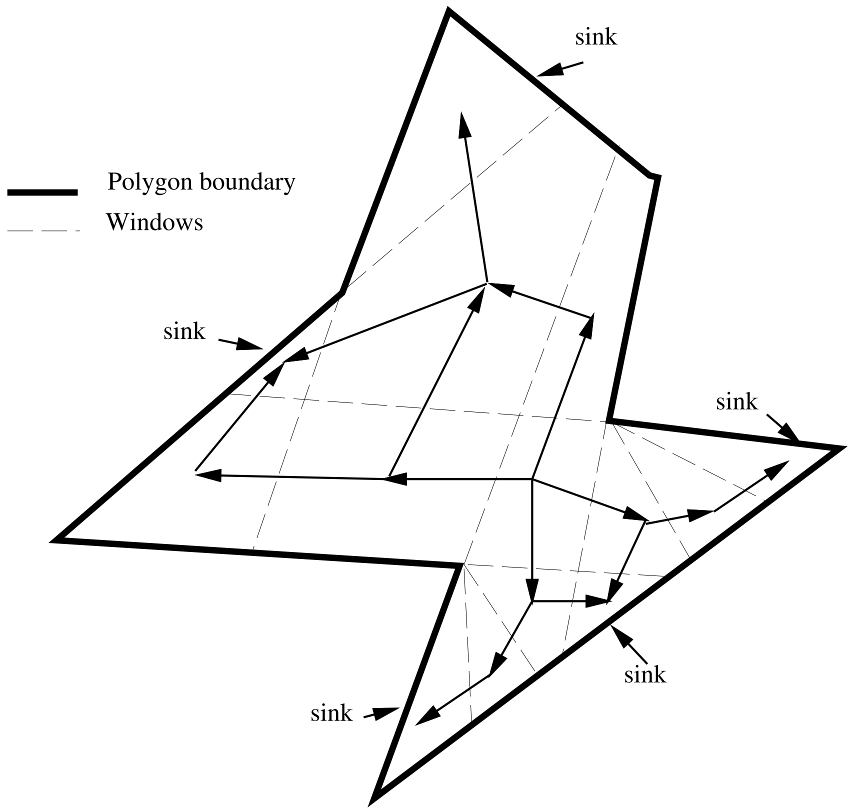

Figure 1: Polygon decomposed into visibility regions with directed dual graph.

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 interior of the polygon P by $int(P)$, the boundary by $bd(P)$, and the exterior by $ext(P)$. The boundary is considered part of the polygon; that is, $P = int(P) \cup bd(P)$. Between any two points s and t on the boundary of 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 $CW(s,t)$ and the counter-clockwise chain by $CCW(s,t)$.

We say a point p is *in* P when $p \in int(P) \cup bd(P)$. 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 $p \in ext(P) \cup bd(P)$. Two points in the *exterior* of a polygon *see* each other if the line segment between them does not intersect the 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 An edge of $VP(x,P)$ that is not contained in an edge of P is called a *window* of point x. Every window of point x is collinear with 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 3. A window is denoted by $w(base, end)$. We refine the definition of windows slightly to eliminate ambiguities that exist due to collinearities. If there is a situation similar to that of (a) and (b) of Figure 2, where a point x is collinear with two reflex vertices of the polygon that lie to one side of x, we say that $w(b, e)$ is the window of point x. If a situation similar to that in (c) arises, we say that there are two windows, one is $w(b, e)$ and the other is $w(b_1, e)$. Finally, if a situation similar to that in (d) arises, we say that there are two windows, one is $w(b, e)$ and the other is $w(b_1, e_1)$. A *pocket* of x in a polygon P is associated with a window of x and defined to be a maximal connected subset of $P \setminus VP(x, P) \cup \{$ the 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 those of the connecting window. A pocket is called

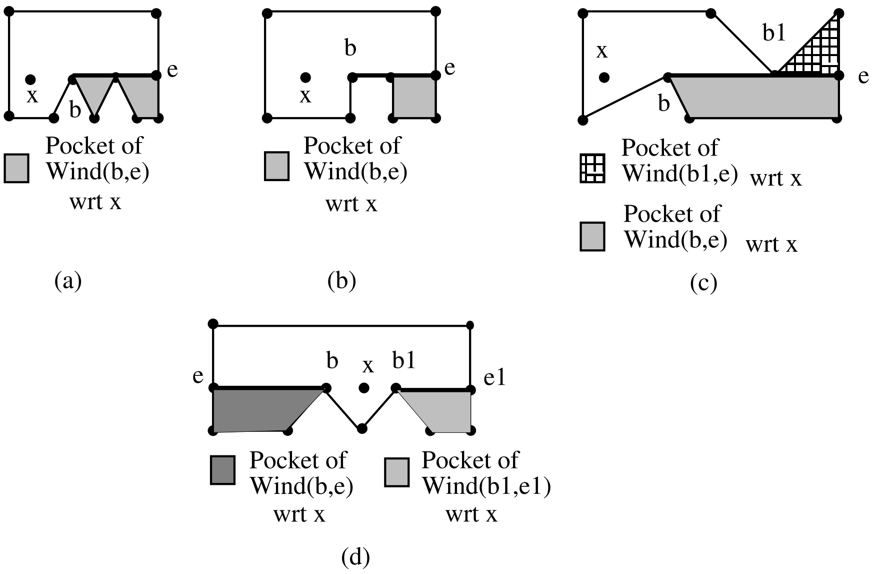

Figure 2: Refined definition of windows

a *left (resp. right) pocket* if in the neighborhood of its window the pocket lies to the left (resp. right) of the ray from x containing the window. A window bounding a left (resp. right) pocket is called a *left (resp. right) window*. The *window half-plane* of left (resp. right) $w(b, e)$ is the left (resp. right) half-plane, defined by the line oriented from b to e. 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. See Figure 3.

Lemma 1 *No point z in a pocket Q of a window w is visible to any point y that is inside w's half-plane but outside Q.*

Proof: Let $w = w(b, e)$. Since $Q \cup w$ is a polygon, the line segment $\overline{zy}$ intersects $Q \cup w$ at least once. As $\overline{yz}$ cannot intersect $\overline{be}$, it intersects some other line segment on the boundary of the

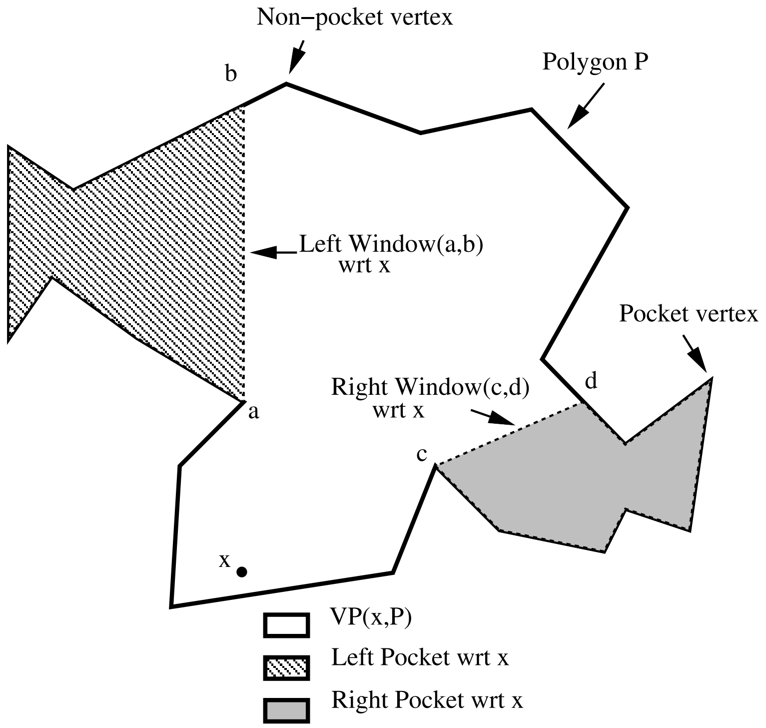

Figure 3: Windows and Pockets

polygon. So, y and z are not visible with respect to the chain Q. ■

3 Visibility Decomposition and its Properties

In this section we prove some key properties of windows and visibility regions that allow us to efficiently store the information necessary for the preprocessing step. Recall from the overview in Section 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 faces of the planar subdivision induced by $W = bd(P) \cup \bigcup_{i=1}^{n} W_i$ are 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 exhibiting 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 obtain our bounds.

Lemma 2 *The base of a window of point x is a reflex vertex of the polygon.*

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 before v in the clockwise order of the vertices and b the vertex occurring after v. The *primary left window* with base v is the left window of a through v and the *primary right window* with base v is the right window of b through v. All of the windows of a point x in a polygon P are collinear with x. Thus, we have:

Lemma 3 *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. Consider the planar subdivision induced by all of the line segments in W. Note that $ext(P)$ is the external face of the subdivision. Unless explicitly stated otherwise, a *face* shall refer to a face of the subdivision that is in P. 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.

Lemma 4 *The set of faces in the subdivision induced by the line segments in W is precisely the set of visibility regions.*

Proof: First, we show that an arbitrary face f in such a subdivision is contained in a visibility region of the polygon. 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 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 but 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 face f. Suppose that a visibility region r was not a face of such a subdivision. Since a visibility region is connected, there must exist two points in r that lie in two adjacent 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 1 *Two visibility regions that share a common edge have the same visibility set except for one vertex.*

Lemma 5 *A visibility region in a simple polygon is convex.*

Proof: Because of Lemma 4, we simply need to show that each face of the planar subdivision is convex. Let a and b be two points in a face f. Suppose that $\overline{ab}$ is not contained in f, then $\overline{ab}$ either intersects a window of some vertex v or an edge $\overline{xy}$ of P. If $\overline{ab}$ intersects a window, then either a or b cannot see v which is a contradiction.

Suppose that $\overline{ab}$ intersects $\overline{xy}$, then there must be a point $z \in ext(P)$ that lies on $\overline{ab}$. If both x and y see the points a and b, this implies that the segments $\overline{xa}, \overline{xb}, \overline{ya}, \overline{yb}$ are wholly contained in P. However, they form a convex quadrilateral and contain the point z contradicting the fact that P is simple. ■

A planar subdivision induced by n^2 arbitrary line segments in the plane can have $O(n^4)$ faces. The following two lemmas use properties of visibility to show that there are only $O(n^3)$ faces in the planar subdivision induced by the $\Omega(n^2)$ line segments in W.

Lemma 6 *Given a line segment $\overline{ab}$ in P, if a point z sees a and b then z sees every 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 7 *At most two windows of a point x intersect a line segment in P.*

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 lemma follows from Lemma 6. ■

Theorem 1 *There are $O(n^3)$ visibility regions in a simple polygon with n vertices and there exist polygons with n vertices that have $\Theta(n^3)$ visibility regions.*

Proof: By Lemma 3, 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 7, we infer that $t < 2n$. Therefore, the total number of intersections in a decomposition is less than $2n^3$. Euler's Formula ([8], 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 4, 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 relies on a bound of $O(n^2)$ regions of minimal visibility or sinks.

Lemma 8 *Given two right pockets, p_1 and p_2, of a point x, no point in p_1 can see a point in p_2.*

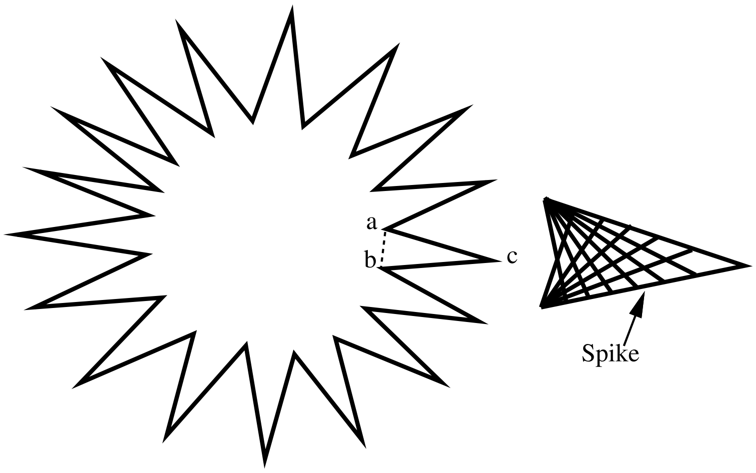

Figure 4: Polygon with $\Theta(n^3)$ visibility regions

Proof: If a point a in p_1 could see a point b in p_2, then the line segment between them must intersect both the window of p_1 and p_2. Consider the window half-planes of p_1 and p_2. Since the two half-planes intersect at x and both windows are right windows, one of the half-planes must contain the window of the other. The lemma follows from Lemma 1. ∎

We now generalize Lemma 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 9 *There is at most one point of intersection between all the right windows of a point x and all the right windows of a distinct point y.*

Proof: Note that if a right window of x intersects a right window of y, then both x and y are visible from the intersection point. Also, from the fact that both are right windows, the base of one window must be contained in the pocket of the other. There are two cases to consider: either x is contained in a right pocket of y or x is not contained in a right pocket of y. We start with the former.

If x is contained in a right pocket of y, then by Lemma 8 x cannot see any other right window of y, and by Lemma 7, only one right window of x can intersect the right window of y's right pocket containing x. Therefore, the lemma follows in this case.

Assume that x is not contained in a right pocket of y. Suppose that a right window r_1 of x intersects a right window r_2 of y. Since x is not contained in a right pocket of y, the base of r_2 must be contained in the pocket of r_1. This implies that y is in the window half-plane of r_1. However, since y is visible from the intersection point of r_1 and r_2, y must be in the pocket of r_1 by Lemma 1. Therefore, the lemma follows since y is contained in a right pocket of x. ∎

Theorem 2 *A decomposition of a polygon with n vertices into visibility regions contains $O(n^2)$ sinks and there exist polygons with n vertices that 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 that is part of a right window in R_i. We will show that there are at most $2n$ such sinks. Let s be a sink 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 sink s 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 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 a vertex v is contained in a right pocket of vertex v_i with window w then by Lemma 8, we know that aside from w, v can see no other window in the set R_i. By Lemma 7, we know that only one left window of v can intersect w. If a vertex v is not contained in a right pocket of vertex v_i, then by Lemma 9, we know there can only be one intersection between the right windows of v and the right windows of v_i. Therefore, there can be at most n sink ends that are part of a left or right window of another vertex.

Since there are at most n right windows in R_i, there can be at most n sink ends that are part of the polygon boundary. Thus, the number of sinks that have an edge that is part of a window in R_i is no more than $2n$. A similar argument shows that the number of sinks that have an edge that is part of a window in L_i is no more than $2n$. Therefore, the number of sinks that have an edge that is part of a window of vertex v_i is no more than $4n$. Since there are n vertices, we conclude that the number of sinks in the polygon is no more than $4n^2$.

The polygon in Figure 5, is a polygon with $\Theta(n^2)$ sinks. The dark squares represent areas that 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. ■

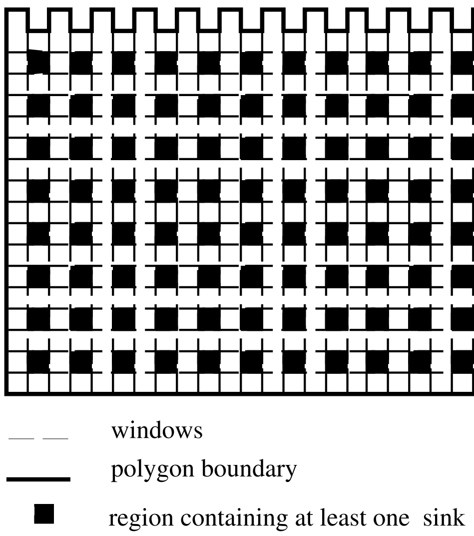

Figure 5: Polygon with $\Theta(n^2)$ sinks, one sink in each dark square.

4 The Visibility Query Algorithm

In this section, we give a high level description of the preprocessing step as described in Section 1.

1. **Construct the set W (as defined in Section 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 [23], Joe and Simpson [21]). 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[14] 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. 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 can be done in $O(n^3 \log n)$ using standard algorithms (Kirkpatrick [22], Lee and Preparata [25], Preparata [28], and Sarnak and Tarjan [30]).
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 takes $O(n^3)$ time. With 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 complexity 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 3 *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.*

5 Recovering the Visibility Polygon

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.

Let us consider two neighboring visibility regions r_1 and r_2. By Lemma 1, we know that the visibility sets of r_1 and r_2 differ by only one vertex, v_1. Without loss of generality, let 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 labeled 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 labeling the window, before the base.

2. If a right window is crossed, insert the vertex labeling the window, after the base.

We prove the validity of this algorithm in the next subsection. The order of the vertices is stored 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 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 $bd(VP(x,P)) \cap bd(P)$. 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 when an end of a window is a vertex. If $w(s,t)$ 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 $w(s,t)$ (see Figure 6 where s and t are vertices 6 and 4, respectively). 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.

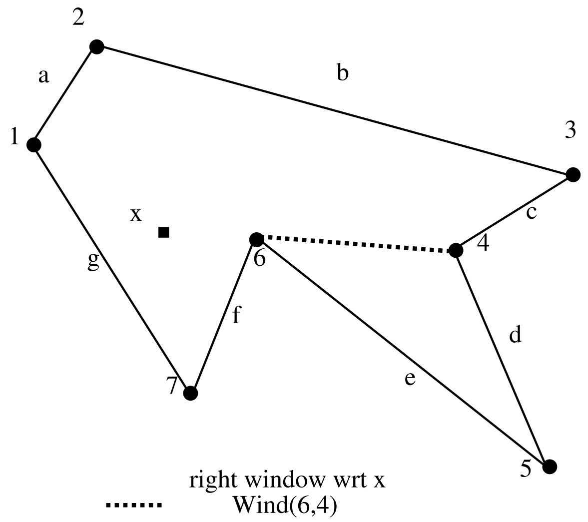

Figure 6: $VS(x,P)$ is 1,a,2,b,3,c,4,d,6,f,7,g

5.1 Recovering the Visibility Sequence

In this subsection, 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 7 serves as an aid to illustrate the lemmas to follow.

Lemma 10 *[23] The clockwise (counter-clockwise) order of the vertices of P around the visibility*

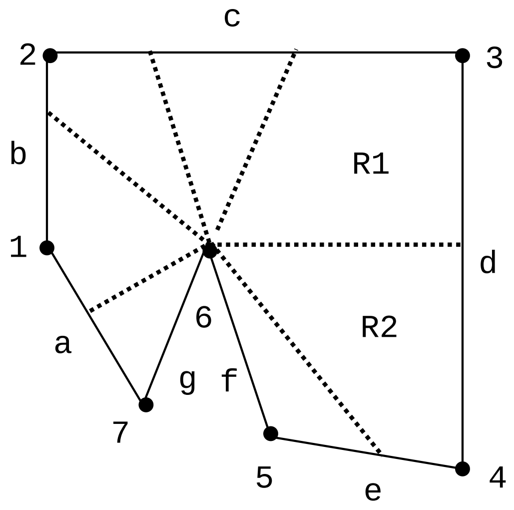

Figure 7: A polygon, decomposed into visibility regions, illustrating the lemmas.

polygon of a point in P preserves the clockwise (counter-clockwise) order of the vertices around P.

We proceed by proving the validity of the simple algorithm described in the previous section. Consider two neighboring visibility regions r_1 and r_2 separated by a right window. Let r_1 be the region that sees one more vertex v. Denote the right window of v separating r_1 and r_2 by $rw(b,e)$ where b is the base and e is the end. We need to determine where to insert v in the ordered visibility set of r_2 to recover the ordered visibility set of r_1. The following lemma shows where the vertex must be inserted.

Lemma 11 *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 $w(b,v) \in VP(p,P)$. ∎

The case for left windows is symmetric.

We now show some properties of a path from a sink to an arbitrary visibility region that will allow us to recover the visibility sequence of a query point.

Lemma 12 *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, $e_1 \in bd(VP(x, P))$ 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 $bd(P)$. 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, $w_2 \in bd(VP(x, P))$ 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 $\overline{p_1 p_2} \in bd(P)$, then the lemma holds. So let us assume that $\overline{p_1 p_2} \notin bd(P)$. But this implies that $CW(p_1, p_2) \in bd(VP(x, P))$ must contain a vertex of P, leading to a contradiction.

All other cases are symmetric. ■

Although, by Lemma 10, 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 13 *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 (note that an edge between two consecutive vertices in a visibility sequence is not necessarily the line segment between them). 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 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 chain $CW(d_1, d_2)$ of

$VP(v_1, P)$ must contain a vertex of P by Lemma 12. 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 $CW(v_1, d_2)$ of the $VP(v_1, P)$ must contain a vertex of P by Lemma 12. 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 $VS(r, P)$. 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.

Lemma 14 *The clockwise ordering of $VS(r_2, P)$ starting at vertex b with the addition of the pair d, v 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 13, 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 11, 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 15 *The counter-clockwise ordering of $VS(r_2, P)$ starting at vertex b with the addition of the pair d, v immediately after b is the same as the counter-clockwise ordering of $VS(r_1, P)$ starting at b.*

We now 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 labeled 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 polygon vertex as well as the polygon edge representing the difference between the two visibility regions.

The labeling 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 14 and Lemma 15, 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 edge pair labeling the window, before the base.

2. If a right window is crossed, insert the edge vertex pair labeling the window, after the base.

We 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 8 shows the structure stored in sink $r1$.

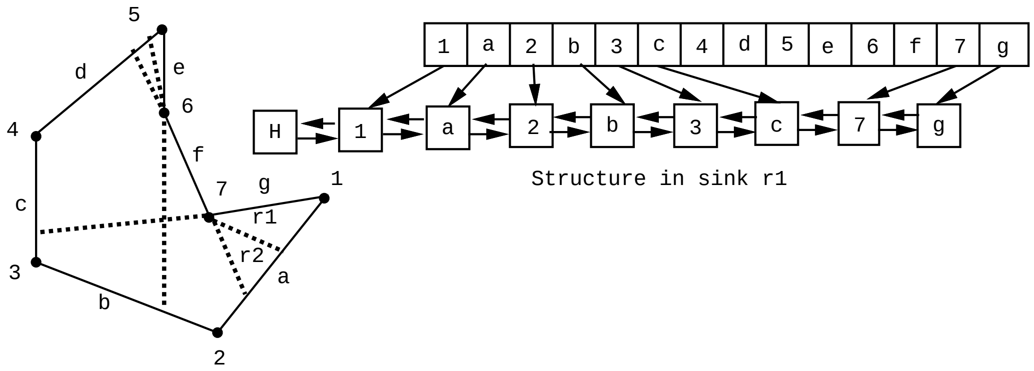

Figure 8: Polygon decomposed into visibility regions and enhanced structure stored in sink.

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 8. 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 15, 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. Figure 9 shows the structure after one update. Since one step requires $O(1)$ time, we conclude with the following.

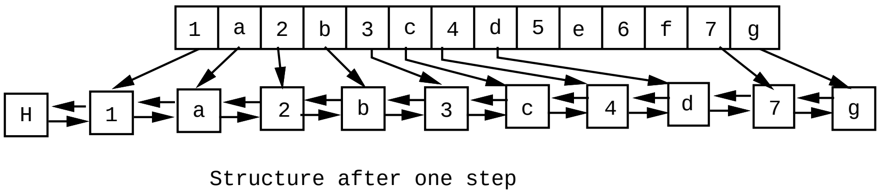

Figure 9: Structure after one update.

Theorem 4 *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 12, 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 polygon 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 *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.*

6 Extensions

6.1 Query points outside the polygon

In this subsection, we extend the solution to handle query points in the *exterior* as well as the interior of the polygon. 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.

Let $CH(P)$ denote the convex hull of polygon P., containing P. An edge of $CH(P)$ that is not an edge of P will be called 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. The following shows that the time and space complexities necessary to preprocess the bays are $O(n^3 \log n)$ and $O(n^3)$, respectively.

Observation 1 *A bay vertex can be in only one bay.*

Observation 2 *A convex hull vertex can be in at most two bays.*

Lemma 16 *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 1 and Observation 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 17 *A convex-hull vertex 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 2, 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 18 *A bay vertex can have at most two of its windows intersecting the outside of the convex hull.*

Proof: By Lemma 2, 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 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 take 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 *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 subsection, 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 subsection 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 be a set of m points labeled $s_1 \dots s_m$. Analogous to the definition of the set W in Section 3, we define a set W'. Let W'_i be the set of all windows in P with respect to point s_i. Let

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

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

Lemma 19 *The set of faces in such a subdivision is precisely the set of visibility regions.*

Proof:

Similiar to proof of Lemma 4.

∎

Corollary 2 *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 7 *With respect to a set of fixed points $\|S\| = m$, a decomposition of a polygon with n vertices into visibility regions contains $O(m^2 n)$ regions, and there exist polygons which have $\Theta(m^2 n)$ visibility regions.*

Proof: By lemma 3, we have

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

Let i be the number of intersections on a single window from the set W'. By lemma 6, we infer that $i < 2m$. Therefore, the total number of intersections in a decomposition is less than $2m^2 n$. Euler's Formula ([8], 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 $4m^2 n$ visibility regions.

The polygon in Figure 10, is a polygon with $\Theta(m^2 n)$ 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. ∎

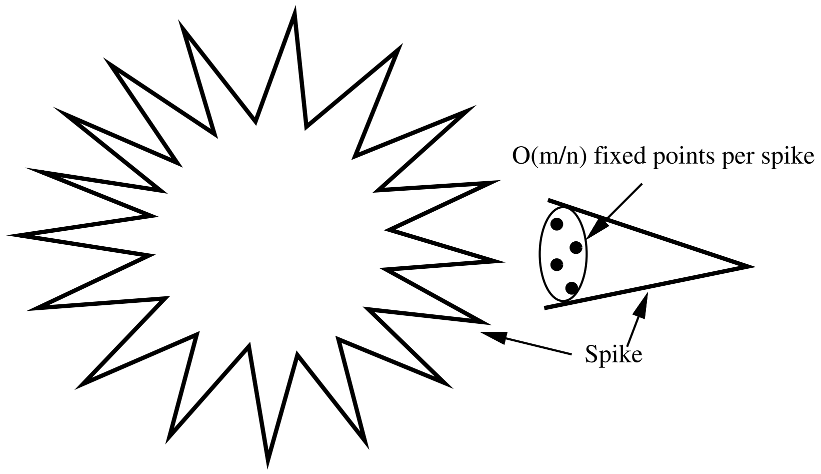

Figure 10: Polygon containing $\Theta(m^2 n)$ visibility regions.

Theorem 8 *With respect to a set of fixed points $\|S\| = m$, a decomposition of a polygon with 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 8, we know that aside from w, s can see no other element of the set R_i. By lemma 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 9, 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 9 *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. The main difficulty seems to be the following. If there are k vertices in the weak visibility set of a line segment, there will be at most $2k$ intersections between the query line segment and the visibility decomposition by lemma 7. However, if there are k vertices in the strong visibility set of a line segment, there can be as many as $2n$ intersections between the line segment and the visibility decomposition. Thus, in the weak visibility case, we are able to remain output sensitive, but in the strong visibility case there does not seem to be a relation between the decomposition and the query line segment that allows us to be output sensitive.

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 10 *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 does not immediately follow from the discussion above and in Section 6.2. The main caveat is that the regions in the visibility decomposition are no longer convex. Thus, finding the path formed by the line segment during a query cannot be done as simply as described above. However, we are able to circumvent this problem by preprocessing each visibility region for ray shooting. Given a simple n vertex polygon, with $O(n)$ preprocessing time and space, we can determine the first intersection point between a query ray and the boundary of the polygon in $O(\log n)$ time (Chazelle and Guibas [12], Guibas et al [17], and Hershberger and Suri [20]). Therefore, since the total complexity of all the visibility regions is $O(m^2(m+n))$, when the set of fixed points has size m and the polygon has n vertices, with $O(m^2(m+n))$ time and space, we can preprocess all the visibility regions for ray shooting queries. Given a query ray in a visibility region, we can in $O(\log mn)$ time recover the first intersection point of the ray with the boundary of the visibility region. Thus, we have

Theorem 11 *A set of m fixed points in the plane and a simple n vertex polygon can be preprocessed in $O(m^2(m+n) \log n)$ time and $O(m^2(m+n))$ space such that given an arbitrary query line segment inside the polygon, the k weakly visible vertices can be recovered in $O(k \log mn)$ time.*

The extension to line segment queries outside the polygon follows from the discussion in Section 6.1.

7 Conclusion

In this paper, 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 the geometric structure underlying the visibility decomposition be used to find efficient solutions to other problems such as the two cover problem (determining whether a polygon is the union of two star-shaped polygons)?

2. Can a non-trivial lower bound be proved for this problem?

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?

References

[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] P. BOSE, A. LUBIW AND J. I. MUNRO. *Efficient visibility queries in simple polygons.* Proceedings of the 4th Canadian Conference on Computational Geometry, St. Johns, Nfld, pp. 23–28, 1992.

[7] P. BOSE. *Visibility in simple polygons.* Masters Thesis, Research report CS-91-61, University of Waterloo, December 1991.

[8] J.A. BONDY AND U.S.R. MURTY. *Graph theory with applications.* Elsevier Science, New York, New York, 1976.

[9] K.W. BOWYER AND C.R. DYER. *Aspect graphs: An introduction and survey of recent results.* Int. J. of Imaging Systems and Technology, 2, pp. 315–328, 1990.

[10] 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.

[11] B. CHAZELLE. *An improved algorithm of the fixed-radius neighbor problem.* Information Processing Letters, **16**, pp. 193-198, 1983.

[12] B. CHAZELLE AND L. GUIBAS. *Visibility and intersection problems in plane geometry.* Discrete and Computational Geometry, **4**, pp. 551-581, 1989.

[13] 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.

[14] 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.

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

[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.* SIAM J. of Computing 26, 4, pp. 1120–1138, 1997.

[19] J. Hershberger. *An optimal visibility graph algorithm of triangulated simple polygons.* Algorithmica, **4**, pp. 141-155, 1989.

[20] J. Hershberger and S. Suri. *A pedestrian approach to ray shooting: shoot a ray, take a walk.* In Proc. 4th ACM-SIAM Symposium on Discrete Algorithms, pp. 54 - 63, 1993.

[21] B. Joe and R.B. Simpson. *Corrections to Lee's visibility polgyon algorithm*, BIT, 27, pp. 458–473, 1986.

[22] D. Kirkpatrick. *Optimal search in planar subdivisions.* SIAM Journal of Computing, **12, 1**, pp. 28-35, 1983.

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

[24] 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.

[25] 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.

[26] K. Mehlhorn. *Multi-dimensional searching and computational geometry.* Springer-Verlag, New York, New York, 1984.

[27] J. O'Rourke. *Art gallery theorems and algorithms.* Oxford University Press, New York, New York, 1987.

[28] F.P. Preparata. *A new approach to planar point location.* SIAM Journal of Computing, **10, 3**, pp. 473-482, 1981.

[29] F.P. Preparata and M.I. Shamos. *Computational geometry, an introduction.* Springer-Verlag, New York, New York, 1985.

[30] N. Sarnak and R. Tarjan. *Planar point location using persistent search trees.* Communications of the ACM, **29, 7**, pp. 669-679, 1986.

[31] T.C. Shermer. *Visibility properties of polygons.* Ph.D. Thesis, McGill University, 1989.

[32] T.C. Shermer. *Recent results in art galleries.* Proceedings of the IEEE, **80, 9**. pp. 1384 - 1399, 1992.

[33] G.T. Toussaint. *Computational geometry.* Elsevier Science, Amsterdam, Netherlands, 1985.