The Robot Localization Problem

Leonidas J. Guibas *
Department of Computer Science
Stanford University
Stanford, CA 94305-2140

Rajeev Motwani †
Department of Computer Science
Stanford University
Stanford, CA 94305-2140

Prabhakar Raghavan
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

July 20, 1995

Abstract

We consider the following problem: given a simple polygon $\mathcal{P}$ and a star-shaped polygon $\mathcal{V}$, find a point (or the set of points) in $\mathcal{P}$ from which the portion of $\mathcal{P}$ that is visible is translation-congruent to $\mathcal{V}$. The problem arises in the localization of robots equipped with a range-finder and a compass — $\mathcal{P}$ is a map of a known environment, $\mathcal{V}$ is the portion visible from the robot's position, and the robot must use this information to determine its position in the map. We give a scheme that preprocesses $\mathcal{P}$ so that any subsequent query $\mathcal{V}$ is answered in optimal time $O(m + \log n + A)$, where m and n are the number of vertices in $\mathcal{V}$ and $\mathcal{P}$, and A is the number of points in $\mathcal{P}$ that are valid answers (the output size). Our technique uses $O(n^5)$ space and preprocessing in the worst case; within certain limits, we can trade off smoothly between the query time and the preprocessing time and space. In the process of solving this problem, we also devise a data structure for output-sensitive determination of the visibility polygon of a query point inside a polygon $\mathcal{P}$. We then consider a variant of the localization problem in which there is a maximum distance to which the robot can "see" — this is motivated by practical considerations, and we outline a similar solution for this case. We finally show that a single localization query $\mathcal{V}$ can be answered in time $O(mn)$ with no preprocessing.

1. Introduction

We consider the following problem: a robot is at an unknown position in an environment for which it has a map. It "looks" about its position, and based on these observations it must infer the place (or set of places) in the map where it could be located. This is known as the *localization problem* in robotics [8, 22].

Aside from being an interesting and fundamental geometric problem, this task has several practical applications. As described in [8], localization eliminates the need for complex position-guidance equipment to be built into factories and buildings. Unmanned spacecraft

*Supported by NSF Grants CCR-9215219 and IRI-9306544, the Stanford OTL fund, the SIMA Stanford Consortium, and the Mitsubishi Corporation.

†Supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Development Award, an OTL grant, NSF Grant CCR-9010517, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

require localization for the following reason [18, 20]: a rover lands on Mars, a map of whose terrain is available to it. It looks about its position, and then infers its exact position on the Martian surface. Another application comes from robots that follow a planned path through a scene: the control systems that guide such a robot along the planned path gradually accumulate errors due to mechanical drift. Thus it is desirable to use localization from time to time to verify the actual position of the robot in the map, and apply corrections as necessary to return it to the planned path [22].

We assume that the robot is in an environment such as an office or factory, with flat vertical walls and a flat floor — thus the problem we address is for polygonal workspaces in two dimensions. The subject of this paper is localization using a *range finder* [19], a device commonly used in real robots [8, 9, 10, 19]. A range finder is a device that emanates a beam (laser or sonic), and determines the distance to the first point of contact with any object in that direction. This is similar to the *finger probe* model [7, 21] studied in computational geometry. In practice a robot sends out a series of beams spaced at small angular intervals about its position, measuring the distance to points at each of these angles. The discrete "points of contact" are then fitted together to obtain a *visibility polygon* $\mathcal{V}$ with m vertices (in general the number of beam probes will be much larger than m).

The robot has a map of its environment: a polygon $\mathcal{P}$ (possibly containing holes) having n vertices. We assume that the robot has a compass: its representations of $\mathcal{P}$ and $\mathcal{V}$ have a common reference direction (say north). We wish to solve the following problem: given $\mathcal{P}$ and $\mathcal{V}$, determine all the points $p \in \mathcal{P}$ such that the visibility polygon of p is translation-congruent to $\mathcal{V}$ — see figure 1

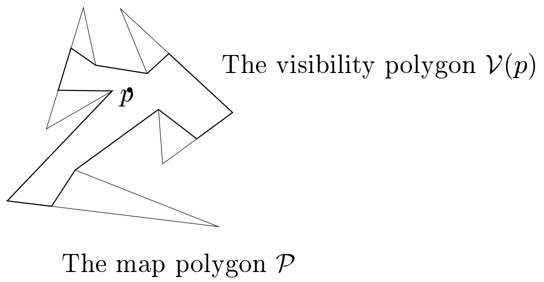

Figure 1: The setting for our robot localization problem

Because the map is likely to be fixed for a given environment, our main interest is in preprocessing the map so that subsequent queries can be answered quickly. Our main contribution is a scheme for preprocessing a simple polygon $\mathcal{P}$ so that any query $\mathcal{V}$ can be answered in time $O(m + \log n + A)$, where A is the size of the output (the number of places in $\mathcal{P}$ at which the visibility polygon would match $\mathcal{V}$); this query time is the best possible. Our preprocessing takes $O(n^5)$ time and space (Section 5). We also exhibit a smooth trade-off between the query time and the preprocessing cost, within certain limits. In these bounds n^5 is really $n^4 r$, where r is the number of reflex vertices in $\mathcal{P}$. Better bounds hold for polygons where the number of mutually visible vertices is smaller than $O(n^2)$ and/or which do not have collinear sides.

The development of our scheme involves the study of a fundamental property of simple polygons — the *visibility cell decomposition* — that has several other applications. Sections 2, 3 and 4 study some properties of this decomposition. An interesting application of these ideas is to the construction of a data structure for the output-sensitive determination of the visibility polygon of a query point (see Section 6.4). This requires a query time of $O(\log n + m)$ using $O(n^2 r)$ preprocessing and space, where m is the size of the visibility polygon. Once again we can trade-off between the query time and the space requirement.

In Section 6 we consider variants of the basic problem. We first describe the effect of holes in the map polygon. Next, we consider a variant motivated by a property of some range-finders — that a distance measurement is obtained only if there is a wall within a certain maximum distance D, and otherwise no reading is obtained (indicating only that the distance to the nearest wall is greater than D in that direction). We show how our approach can be modified to deal with this feature without increasing the query time, but with additional preprocessing. We then address the following question: given no preprocessing, what is the complexity of answering a single query? We provide an algorithm running in time $O(mn)$ by applying results on ray-shooting.

Independently of our work, Bose, Lubiw and Munro [2] obtained some of the results presented here. In particular, they provide a scheme for preprocessing a simple polygon so as to compute the vertices of the polygon that are visible from a query point in time $O(\log n+m)$ time. The machinery developed for this purpose includes some of the visibility cell decomposition structure theorems described below.

1.1. Overview of our scheme

We now give a brief overview of our scheme to motivate the study of the visibility cell decomposition in Sections 2, 3 and 4. We approach the problem by partitioning the map polygon into regions such that within a region the visibility polygon of any point is roughly the same — in Section 2 we call this rough view a *skeleton*. An intuitive definition of the skeleton of $\mathcal{V}$ is that it is a contraction of $\mathcal{V}$ so that the skeleton boundary contains exactly those vertices from $\mathcal{V}$ that can be certified to be vertices of $\mathcal{P}$. We provide a data structure which quickly identifies all the regions that have the same skeleton as the query $\mathcal{V}$. We then check the candidate regions to see if they contain any points that have exactly the same view as $\mathcal{V}$. Some difficulties that arise are:

a) Due to occlusions by reflex vertices, an edge of the map polygon may have neither, or only one, of its end-points visible from a point inside the polygon. Our characterization of a skeleton must cater to these incomplete edges.

b) If the line segments forming several edges of the polygon are collinear, it is possible that a "window" in the map allows the robot to see only an interior portion of one of these edges. Further, it cannot easily identify which of these collinear edges it sees. The problem is compounded when there are several such windows and collections of collinear edges. In fact, this is one source of complexity in our preprocessing. In the case where the map $\mathcal{P}$ has no collinear edges the preprocessing space can be improved to $O(n^3 r)$ — see Section 5.

c) There can be regions that match the skeleton but contain no point whose visibility polygon is congruent to V. Thus we must still pinpoint those visibility regions (from all the ones that share this skeleton) that contain a point whose visibility polygon exactly matches $\mathcal{V}$. We must do so in time proportional to A, so we cannot check each candidate region individually. We reduce this problem to a form of point-location in a planar subdivision.

2. Visibility Polygons and Skeletons

Let $\mathcal{P}$ denote a polygon with n sides. We will refer to $\mathcal{P}$ as the map polygon. Let P denote the boundary of $\mathcal{P}$. We first assume that $\mathcal{P}$ has no holes, deferring this general case to Section 6.1. From now on all polygons will be assumed to be oriented with respect to a common reference direction.

Two points in P are visible to each other if the straight line joining them meets P only at these end-points. The visibility polygon $\mathcal{V}(p)$, for any point $p \in \mathcal{P}$, is the polygon consisting of all points in $\mathcal{P}$ that are visible from p. Assume that the number of vertices in $\mathcal{V}(p)$ is m.

Let $V(p)$ denote the boundary of $\mathcal{V}(p)$. Assume that p does not lie on P, and hence it does not lie on $V(p)$. In general, there will be edges and vertices in $V(p)$ which do not coincide with edges and vertices in P. To deal with such cases we define the notion of spurious edges and vertices. Informally, an edge or a vertex is non-spurious if the view from p provides a guarantee that this edge or vertex is on the boundary of $\mathcal{P}$.

Definition 1: *An edge of $V(p)$ is* spurious *if it is collinear with p.*

Definition 2: *A vertex $v \in V(p)$ is* spurious *if it lies on a spurious edge $(u, v) \in V(p)$ and the other end-point u is closer to p.*

Some of these notions are illustrated in figure 2.

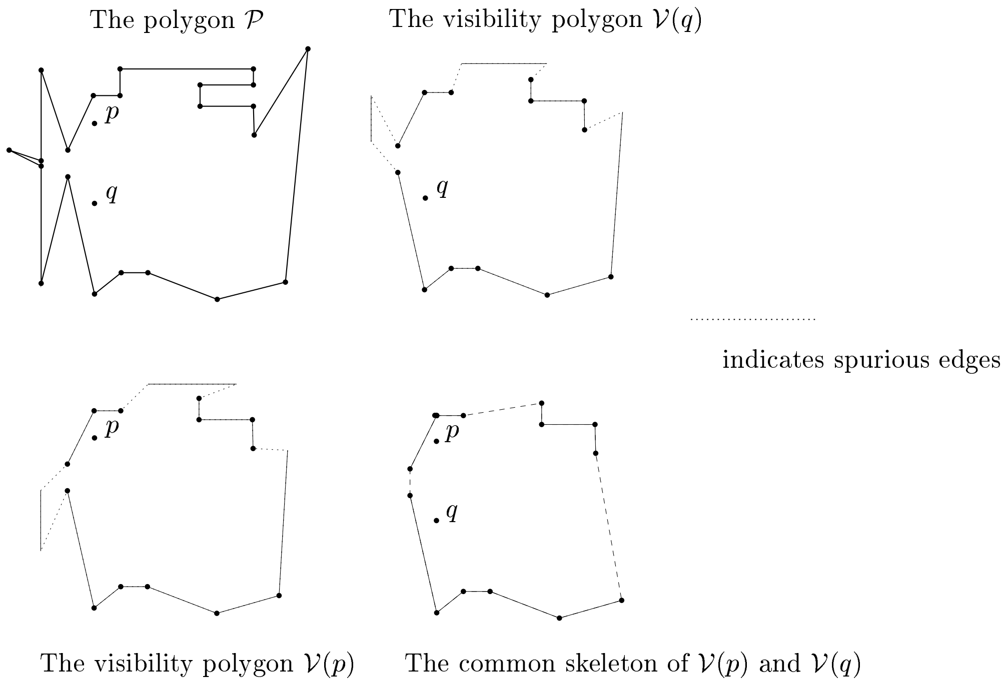

Figure 2: Visibility polygons and their skeletons

This definition may label as spurious an edge (or a vertex) which actually lies on P. This only happens if that edge is collinear with p. In that case, the closer of the two end-points of the edge may be visible from p but it will then block the view of any other point on the edge. Thus, although the edge (u, v) is in $V(p)$, the robot sitting at point p cannot infer this from its localized view. Similarly, the definition of a spurious vertex assumes that if a ray from p goes through vertices u and v of P, in that order, then u is an obstacle to the visibility of v from p. As the next lemma shows, the non-spurious components of $V(p)$ are invariant under any modifications to $\mathcal{P} \setminus \mathcal{V}(p)$. The proof is an easy consequence of the above definitions.

Lemma 1: *An edge e, or a vertex v, in V is non-spurious if and only for each choice of $\mathcal{P}$ and $p \in \mathcal{P}$ such that $\mathcal{V}(p) = V$, e and v lie on P.*

A reflex vertex in P is a vertex which subtends an angle greater than 180^0 inside $\mathcal{P}$. It is the existence of reflex vertices which creates obstacles to viewing the points inside $\mathcal{P}$. Note that a spurious vertex can never be a reflex vertex in $V(p)$.

Definition 3: *A reflex vertex v in $V(p)$ is a* blocking vertex *if at least one edge incident on v in P does not intersect $\mathcal{V}(p)$.*

It is now easy to establish the following lemma.

Lemma 2: *If a vertex in $V(p)$ is a non-blocking reflex vertex, then both its incident edges in $V(p)$ must be non-spurious.*

The next lemma follows from the observation that each spurious edge can be extended to pass through p. Thus, no two spurious edges of $V(p)$ can meet each other except at p, and by assumption p does not lie on the boundary of $\mathcal{V}(p)$.

Lemma 3: *No two spurious edges can be adjacent in $V(p)$.*

For each spurious edge, the end-point closer to p is a blocking reflex vertex and the other end-point is a spurious vertex.

Lemma 4: *Let $e \in V(p)$ be a non-spurious edge and e' the edge of P on which it lies. Then e is the only portion of e' visible from p and the edge e is of one of the following three types.*

- full edge*: the end-points of e are the same as those of e';*
- half edge*: one end-point of e is spurious, the other is an end-point of e';*
- partial edge*: both end-points of e are spurious vertices.*

We now conclude that the spurious vertices and edges in $V(p)$ can only occur in certain specific patterns. Consider a clockwise traversal of $V(p)$ starting with an arbitrary blocking (and therefore reflex) vertex. (If no such vertex exists then $\mathcal{V}(p) = \mathcal{P}$, trivializing the whole problem, so from now on we assume the existence of such a vertex.) The sequence of vertices seen in this traversal can be decomposed into chains of consecutive non-spurious vertices alternating with chains of consecutive spurious vertices — call this the vertex chain decomposition of $V(p)$.

Lemma 5: *The* vertex chain decomposition *of $V(p)$ has the following properties.*

1. *A non-spurious chain can contain blocking vertices only as its end-points.*
2. *A spurious chain is of length at most two.*

3. *Consider a spurious chain with only one vertex v. Let x be the last vertex of the preceding chain and y the first vertex of the succeeding chain. Then one of x and y is a blocking vertex joined to v by a spurious edge, while the other is non-blocking and is joined to v by a half-edge.*

4. *Consider a spurious chain with two vertices u and v, in that order. Let x be the last vertex of the preceding chain and y the first vertex of the succeeding chain. Then both x and y are blocking vertices with spurious edges going to u and v, respectively, and the edge (u, v) is a partial edge.*

Fix a canonical blocking reflex vertex v_o of $V(p)$ as the origin with reference to which we specify all other points. For example, v_o can be the leftmost blocking reflex vertex of $V(p)$. Let $V^*(p)$ be the polygon induced by the non-spurious vertices of $V(p)$ ordered by a clockwise traversal of starting at v_o. In $V^*(p)$, if two adjacent vertices are from the same chain in $V(p)$, then their edge is the same as in $V(p)$, and this must be a non-spurious edge. Otherwise, the two vertices are end-points of neighboring chains in $V(p)$ and their edge in $V^*(p)$ is a newly introduced artificial edge.

Every artificial edge e' of $V^*(p)$ corresponds to some half or partial edge e of $V(p)$. The edge e is one of the edges of $V(p)$ which connect the two chains whose end-points are the vertices of e'. We will label each artificial edge e' with a characterization of the line on which the corresponding edge e lies. This line-characterization will be the coefficients of the linear equation that defines the line containing e, with the origin at v_o.

Definition 4: *The* skeleton $V^*(p)$ *of a visibility polygon $\mathcal{V}(p)$ is the polygon induced by the non-spurious vertices of $V(p)$. Each artificial edge of $V^*(p)$ is labeled with the line equation and the type of the corresponding half or partial edge.*

The skeleton of a visibility polygon can also be looked upon as a polygon induced by all the full edges in $V(p)$, such that the chains of edges are tied together by artificial edges. It is important to keep in mind that a skeleton is a *labeled* polygon as described above.

Definition 5: *The* embedding *of a skeleton $V^*(p)$ is a 1-1 mapping h from the vertices in the skeleton into the vertices of P such that:*

1. *For each vertex v in $V^*(p)$, the location of $h(v)$ relative to $h(v_o)$ is identical to the location of v relative to v_o in $V(p)$.*

2. *There is a full edge between vertices u and v in $V^*(p)$ if and only if there is an edge of P with $h(u)$ and $h(v)$ as end-points.*

3. *Let l' be the line labeling an artificial edge between vertices u and v in $V^*(p)$. Then there is an edge e of P lying on a line l whose equation (with $h(v_o)$ as origin) is that of l', and a point of e visible from both u and v.*

Does $V^*(p)$ have enough information to uniquely determine the point p? Unfortunately not: the information about the end-points of a half or partial edge e in $V(p)$ is absent from the labels of the corresponding edge e' in $V^*(p)$. (The reason for this imprecise labeling will become clear later when we describe our search mechanism.) Thus a single embedding may have several candidate edges in P for the edge e. These candidate edges must all be collinear. For instance, in figure 2, the partial edge on the left visible from p, and that visible from q, are collinear. Note that the skeletons of p and q come out to be the same.

Let r denote the number of reflex vertices in $\mathcal{P}$.

Theorem 1: *Given a visibility polygon $\mathcal{V}(p)$, its skeleton $V^*(p)$ has at most r embeddings in $\mathcal{P}$ and this bound is best possible.*

Proof: Let us label any embedding h of $V^*(p)$ in $\mathcal{P}$ by the vertex $h(v_o)$, where v_o is the origin vertex in $V^*(p)$. We claim that there can be at most one embedding with the label v, for any vertex v in P. This follows from the observation that the location of every vertex in $V^*(p)$ is fixed with reference to the origin vertex. Having specified the location of the origin as being at v immediately fixes the location of every other vertex and thus uniquely determines the embedding h itself.

Since v_o is a reflex vertex, there are at most r distinct labels for the embeddings of $V^*(p)$. It follows that number of distinct embeddings cannot exceed r.

To see this bound is tight to within constant factors, consider the polygon $\mathcal{P}$ and the visibility polygon $\mathcal{V}(p)$ in figure 3. It is clear that the origin of $V^*(p)$ can be mapped to any of the vertices marked with o in P. Clearly, there are $\Theta(r)$ such embeddings. □

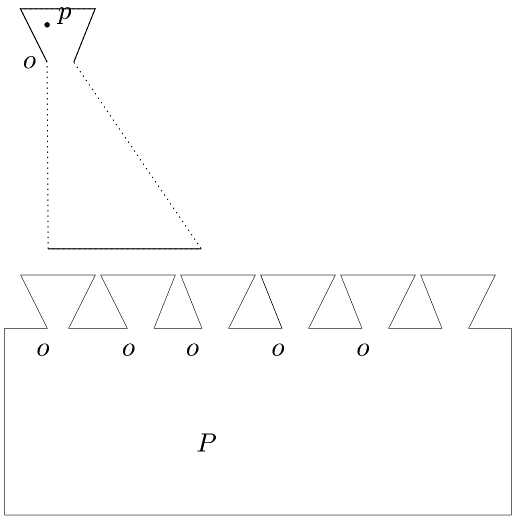

Figure 3: A bad case for localization

The location of the point p is fixed with reference to the origin vertex v_o of $V^*(p)$. Thus, the only possible locations of the point p are the (at most r) locations corresponding to the different embeddings of the skeleton. Notice that an embedding of $V^*(p)$ may not correspond to an embedding of $V(p)$, although the converse is always true.

Corollary 1: *The number of solutions in $\mathcal{P}$ to a localization query $\mathcal{V}$ is at most r.*

3. Visibility Cell Decomposition

We now describe the subdivision of the map polygon into visibility cells such that the points in each cell have essentially the same visibility polygon. The subdivision is created by introducing lines into the interior of the map polygon. Each line partitions $\mathcal{P}$ into two regions, one where a vertex v is not visible due to the obstruction created by vertex u and another region where u cannot block the view of v. Each such line starts at a reflex vertex

u and ends at the boundary of $\mathcal{P}$. It is collinear with a vertex v which is either visible from u or is adjacent to it in P. This line is said to be *emanating* from v and *anchored* at u.

It is convenient to give each of the interior lines a direction and consider the interior of the map polygon to be dissected by a collection of such rays. A ray determined by vertex v as emanating vertex and vertex u as anchor vertex proceeds *from* u into P *away* from v. It forms the boundary between regions that can locally see v and others that cannot. We will classify this ray as a *left* or *right* ray for v according to whether the obstruction defining the anchor u is to the left or the right of the ray, from the point of view of an observer sitting on v and looking along the ray. Note again that a ray starts at the anchor vertex and proceeds away from the emanating vertex.

Theorem 2: *In the visibility cell decomposition of the map polygon $\mathcal{P}$:*

1. *The number of lines introduced in the interior of $\mathcal{P}$ is $\mathrm{O}(nr)$, and this bound is the best possible.*
2. *Each cell in the decomposition is a convex polygonal region inside $\mathcal{P}$.*

Proof: It easy to see that there are at most $\mathrm{O}(nr)$ interior lines in the decomposition since each interior line is generated by a pair of anchor and emanating vertices, and each pair of vertices generates at most two interior lines. It is also quite easy to construct examples where the upper bound is achieved.

For the second part, let $\mathcal{C}$ be a visibility cell which is non-convex and let w be a reflex vertex on the boundary of $\mathcal{C}$. We first claim that w cannot be a vertex from the boundary of $\mathcal{P}$. Otherwise, the edges incident on w would be extended into interior lines, and these interior lines would subdivide $\mathcal{C}$. On the other hand, no interior vertex can be a reflex vertex for its bordering visibility cells since it is formed by the intersection of two interior lines which start and end at the polygon boundary. This gives a contradiction. $\square$

Since we can have $\Theta(nr)$ lines forming the cell decomposition, an obvious bound on the total complexity of the cells in the decomposition is $\mathrm{O}(n^2r^2)$. However, the structure of our problem can be exploited to obtain the following tighter bound.

Theorem 3: *The number of visibility cells in a given map polygon, as well as their total complexity, is $\mathrm{O}(n^2r)$, and this bound is best possible.*

Proof: We will show that the number of subdivision vertices is $\mathrm{O}(n^2r)$. There is a total of $\mathrm{O}(nr)$ rays, and each ray gives rise to one boundary vertex only. Therefore it suffices to count non-boundary vertices for the asymptotic bound in the theorem.

The simplest proof of this theorem is consists of considering each of these $\mathrm{O}(nr)$ rays separately. Let R be such a ray in the subdivision, anchored at A and emanating from a. Suppose we walk along this ray, starting at A and going away from a. During this walk there can be at most $2n$ changes in the set of visible vertices. This is because, in a simple polygon, once a vertex disappears from view when we walk in the polygon along a straight line, it will never become visible again (otherwise the occluder in-between represents a hole in the polygon). This implies that the ray R has $\mathrm{O}(n)$ vertices on it. Since the number of rays is $\mathrm{O}(nr)$, we get the desired bound.

However, we also present now a more involved proof, whose structural aspects will be useful in arguments later on.

Consider a vertex of the subdivision which does not lie on the boundary P. There must be two rays whose intersection gives rise to this vertex v. Let the first ray emanate from the vertex a with the anchor vertex A, and the second ray emanate from b with the anchor B. Consider the two lines containing the given rays, they divide the plane into four "quadrants." Notice that if both rays are of the same orientation (i.e. both left or both right) then the corresponding anchors must lie in adjacent quadrants. In the case where the two rays have different orientations, the corresponding anchors lie in either the same quadrant or in opposite quadrants.

Label each vertex of the subdivision that does not lie on the boundary of P by the pair of vertices that the two rays determining the vertex emanate from, as well as by the two bits specifying the relative placement of the two anchors in the quadrants defined above. The two bits classify the subdivision vertices into four types. Type 1 vertices are determined by two left rays and have the anchors in adjacent quadrants; Type 2 vertices are analogous except that the rays are both of the right orientation. Type 3 vertices have the anchors in the same quadrant, while Type 4 have the anchors in opposite quadrants. Figure 4 illustrates a Type 4 vertex.

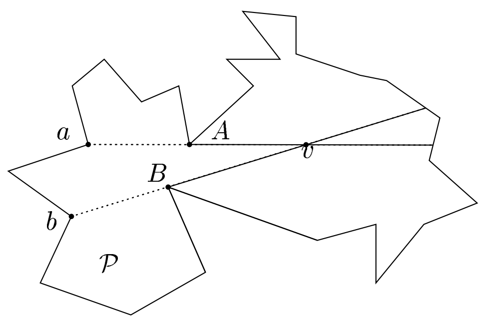

Figure 4: A Type 4 vertex of the visibility decomposition

The hardest case to deal with is that of the Type 4 vertices — when the two anchors lie in opposite quadrants. Consider a Type 4 vertex v determined by two rays, say a ray from a with right anchor A, and a ray from b with left anchor B. Suppose we take the portion of the boundary of P from A to B not containing a or b and replace it with the interior segments Av and vB. We have not eliminated any vertices of our subdivision with the same label as v because in the eliminated part of P at least one of a of b is not visible. In the remaining part of $\mathcal{P}$, there cannot be any other Type 4 vertex labeled by a and b which uses either A or B as an anchor for the two rays. Thus, there can be at most r vertices with this label. Since the total number of labels is $O(n^2)$, we have the desired bound.

There are three other cases to consider. In each case, a simple topological argument shows that only a *unique* vertex can possess that label. See for example figure 5, which shows the situation for two left rays. Thus there can be a total of $O(n^2 r)$ interior vertices of our subdivision and so the number of cells is similarly bounded.

For a lower bound, consider a polygon P that has $\Omega(r)$ small bays (each say of three sides) lying roughly along a straight line and facing a convex chain of $\Omega(n)$ sides that is visible from all of them — see figure 6. Then within each bay we can get $\Theta(n^2)$ regions corresponding to the visible subchain of the convex chain. This gives a total of $\Theta(n^2 r)$ subdivision vertices in total. $\square$

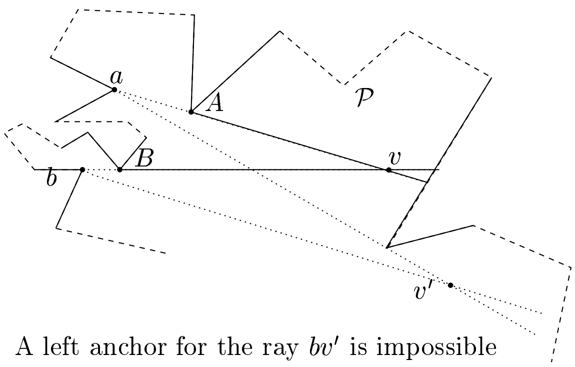

Figure 5: Labels with two left anchors are unique

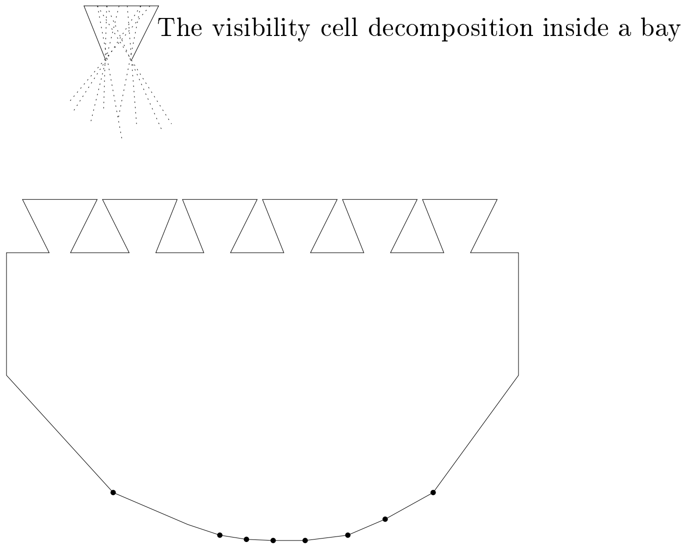

Figure 6: A polygon with a large visibility cell decomposition

4. Visibility from a Cell

We now examine the visibility polygons for all the points in a particular cell and extract some common features from these views. We must be careful about the assignment of the points on the interior lines to the cells in the decomposition. Consider an interior ray emanating from vertex u and anchored at vertex v. The cells bordering this ray are divided into classes, those which can see u and those which cannot. The boundary edges determined by this ray are assumed to be a part of the latter kind of cells only. Using this rule, each interior vertex gets assigned to a unique cell as well.

The following theorem ties together the notions of a skeleton and a visibility cell.

Theorem 4: *For any visibility cell $\mathcal{C}$, and points p, $q \in \mathcal{C}$, $V^*(p) = V^*(q)$.*

Proof: Consider the straight line joining p and q, call this line l. Clearly, l is totally contained in $\mathcal{C}$, even when p and q are boundary points. Further, no interior line of the decomposition intersects l. Suppose s is the point on l which is the closest to q such that it has a different visibility skeleton than $V^*(q)$.

Consider first the case where $V^*(s)$ and $V^*(q)$ do not have the same underlying polygon, i.e. they have a different set of vertices. Assume, without loss of generality, that the difference between the two skeletons is that a vertex of P, say x, is visible from s but is not visible from q. Then there must exist a reflex vertex $y \in V^*(q)$ such that it is an obstruction for q viewing x. Then the ray emanating from x and anchored at y must intersect l between s and q, giving a contradiction. Therefore, the underlying polygon for the skeleton is invariant over the entire cell.

Any difference in the skeletons $V^*(q)$ and $V^*(s)$ must then be in the labeling of the edges. It is fairly easy to verify that the location of full edges and artificial edges must be the same in both cases, as also the labeling of the artificial edges as corresponding to either half or partial edges. The difference, if any, must be in the line-characterizations that label some artificial edge. In that case, there must exist two edges of P, say e_s and e_q, such that s can see some portion of e_s but cannot see e_q at all, and q can see some portion of e_q but cannot see e_s at all. We then conclude that there is some point between s and q on l which sees a vertex not visible from q. This contradicts the assumption that s was the closest point of l to q which has a different skeleton.

Thus, there can be no point s on l which has different visibility skeleton from q. $\square$

This theorem allows us to make the following definition.

Definition 6: *For any visibility cell $\mathcal{C}$, the* visibility skeleton of the cell $V^*(\mathcal{C})$ *is the common visibility skeleton for all points contained in $\mathcal{C}$.*

The exact choice of an edge e of P for any half or partial edge is also invariant over the cell, although the portion of e that can be seen varies from point to point in the cell. However, Theorem 4 does not guarantee that if two points have the same visibility skeleton then they are in the same visibility cell. In fact, a visibility skeleton can have r embeddings and could have several distinct visibility cells in its kernel, all with the same skeleton (as was already discussed — see figure 2). This is because there could be several collinear edges of P that are all candidates for being the half or partial edge corresponding to a particular artificial edge of the skeleton. If we assume that P contains no collinear edges, then there can be only one relevant visibility cell in the kernel of each embedding of the skeleton.

Definition 7: *The binary relation "$\equiv$" over the visibility cells in the decomposition is such that for any two cells $\mathcal{C}_1 \equiv \mathcal{C}_2$ if and only if $V^*(\mathcal{C}_1) = V^*(\mathcal{C}_2)$.*

It is easy to verify that this is an equivalence relation over the cells such that each equivalence class of cells is associated with a unique visibility skeleton. Let $\mathcal{E}_i$, for $1 \leq i \leq t$, be the equivalence classes for $\mathcal{P}$, and denote by V_i^* the visibility skeleton of the cells in the i^{th} equivalence class. Let also e_i denote the total size of all visibility cells in $\mathcal{E}_i$.

Before we embark on explaining our searching data structures, we would like to characterize the complexity of a visibility cell in terms of the complexity of its visibility skeleton. Recall that each side of a cell arises from a ray anchored at an end-point of an artificial edge.

Theorem 5: *Let $\mathcal{C}$ be a visibility cell whose visibility skeleton has s artificial edges. Then the complexity of $\mathcal{C}$ is $\mathrm{O}(s)$.*

Proof: Each side of the polygon $\mathcal{C}$ is determined by a ray anchored at one of the blocking reflex vertices. Each such blocking vertex can have at most two rays which bound $\mathcal{C}$. Since the number of blocking reflex vertices is $\mathrm{O}(s)$, the result follows. □

5. Data Structures and Search Algorithms

We now describe the construction of the data structures and algorithms for query processing. In the first step of preprocessing we compute the visibility cell decomposition of the map polygon, and the visibility skeleton for each cell. Each skeleton polygon of size s is represented as an $\mathrm{O}(s)$-dimensional real vector. These vectors are stored in a multi-dimensional search tree each of whose leaves indexes an equivalence class of cells with the same skeleton. Given $\mathcal{V}(p)$, we extract the visibility skeleton $V^*(p)$ and query this data structure to identify the equivalence class of cells where p must be located. The last, and most non-trivial, stage of the search is concerned with identifying the possible locations of p within the equivalence class. This reduces to a search problem in a planar subdivision.

5.1. Computing Cells and their Skeletons

The visibility cells and their skeletons are computed by the following steps: (i) for each reflex vertex, identify the vertices of $\mathcal{P}$ that are visible from it — each such vertex can give rise to a line in the arrangement with one end-point at that reflex vertex; (ii) compute the arrangement of all these lines inside $\mathcal{P}$; (iii) compute, for each cell in the arrangement, the visibility polygon (and hence the skeleton). This last computation can be done in an incremental fashion as we walk along a line of the arrangement — the visibility polygon incurs only one change from one vertex on a line to the next vertex. Let C denote the number of visibility cells and N the number of visibility cell vertices; note that C and N are always $\mathrm{O}(n^2r)$.

Theorem 6: *The preprocessing time and space for computing the visibility cell decomposition are* $\mathrm{O}(nr\log n + nN)$.

Proof: By standard results in ray-shooting in simple polygons [15] and the construction of arrangements via sweepline methods [11], steps (i) and (ii) above can together be completed in $\mathrm{O}(nr\log n + N\log n)$ time and $\mathrm{O}(nr + N)$ space. To bound the complexity of step (iii), we note that the visibility skeletons of adjacent regions differ in at most a constant number of contiguous edges. Thus we may generate the C visibility skeletons by the following procedure that walks along the lines forming the arrangement. We assume for simplicity of description that there are no degeneracies in the arrangement, i.e., at most two lines intersect at any point of the arrangement except at vertices of $\mathcal{P}$. Assume that every vertex of the arrangement that is not a vertex of $\mathcal{P}$ is labeled by the emanating and anchor vertices of the two lines that intersect to form that vertex.

We first compute the visibility polygons of the r reflex vertices in time $\mathrm{O}(nr)$ [14]. Call this set ρ.

We begin at the (reflex) anchor vertex of the first line (call it ℓ) in the arrangement; given ρ, we can in time $\mathrm{O}(n)$ obtain the visibility skeleton of the cell of the arrangement that contains ℓ and its anchor and lies to the right of ℓ. We then walk on the arrangement along ℓ; at each new vertex w of the arrangement a vertex v of $\mathcal{P}$ either becomes visible or invisible. Indeed, v is the emanating vertex of the other line that intersects ℓ at w. From this we can in $O(n)$ time and space read off and write down the visibility skeleton of the region bounded by line ℓ, vertex w, and lying to the right of ℓ. We repeat this for all the lines forming the arrangement, for a total of $O(Nn)$ time and space.

By combining the complexities of all three steps, the result follows. $\square$

Let $\mathcal{R}_i$, $1 \leq i \leq C$, be the regions/cells in the subdivision. Define r_i to be the complexity of $\mathcal{R}_i$, m_i to be the complexity of the $V^*(\mathcal{R}_i)$ and s_i to be the number of blocking vertices in these skeletons. Note that r_i is $O(s_i)$, and that $s_i \leq m_i$.

5.2. Locating the Equivalence Class

Assume that we have obtained a subdivision of $\mathcal{P}$ into C visibility cells, and have computed a visibility polygon for one point in each of the cells. We start by showing how to compute the equivalence classes.

Given a visibility polygon of complexity m, the corresponding skeleton can be computed in $O(m)$ time. We will encode each of the C skeletons as an M-dimensional real vector, where $M = O(m)$. The encoding fixes an origin of the skeleton at a canonical reflex vertex (as described earlier) and specifies the position of every other vertex relative to the origin using $2m$ real numbers. Further, we store the edge labels using another $2m$ components. The ordering of the vertices and edges in the skeleton is specified implicitly by the ordering of the components of the encoding. This entire process takes $O(m)$ time.

A crucial property of the representation is that two cells have the same skeleton if and only if their representations are identical. This motivated the definition of a skeleton and is vital to constructing and searching the equivalence classes.

We first partition the cells according to the number of vertices in their visibility skeletons. Consider now the T_s cells whose skeletons have size s. We construct a multi-dimensional search tree [17] for the vectors corresponding to the skeleton representations of these cells. These search trees can be constructed in $O(sT_s + T_s \log T_s)$ time and space, and support exact match queries in time $O(s + \log T_s)$. The various skeletons whose vector representations are identical will reach the same leaf of this search tree. Thus the leaves are in 1-1 correspondence with the equivalence classes of cells in this collection. (In practice, it would be more efficient to recompute the equivalence search trees using one representative from each equivalence class, after having computed the equivalence classes as above.)

We have at most n different search trees corresponding to the different values of s. Given a query $\mathcal{V}$, we can easily determine which tree to search and thence the correct equivalence class.

Theorem 7: *The t equivalence classes, $\mathcal{E}_i$, can be computed in $O(nN)$ time using the equivalence search trees. The n search trees can all be constructed in $O(nN)$ time and space, and answer queries in time $O(s + \log n)$, where s is the query skeleton size.*

5.3. Searching within an Equivalence Class

It remains to specify how we search within an equivalence class of visibility cells for all the possible locations of the query point p. We will have one data structure for each of the at most n equivalence classes, associated with the corresponding leaf of the equivalence tree.

We now fix our attention on any one equivalence class $\mathcal{E}_i$. Let V^* be the skeleton corresponding to $\mathcal{E}_i$. Each cell in $\mathcal{E}_i$ can be identified with a distinct embedding of V^*. The cell must lie in the kernel of that embedding. For each embedding there could be several cells in the kernel, but these must be disjoint convex polygonal regions (recall that there will be only one cell per embedding if P contains no collinear sides). Let the class $\mathcal{E}_i$ consist of cells from k different embeddings. The complexity of any one cell in $\mathcal{E}_i$ is at most m, the size of $\mathcal{V}(p)$. The following theorem bounds the overall complexity e_i of all the cells in $\mathcal{E}_i$.

Theorem 8: *The total number of cells in any equivalence class $\mathcal{E}_i$, as well as their total complexity e_i, is $\mathrm{O}(n^2)$. This bound is best possible.*

It turns out to be much easier to prove a stronger version of the above theorem.

Definition 8: *The class $\mathcal{C}_m$ of cells in the decomposition consists of all cells whose skeletons contain m non-spurious vertices.*

We will show that the total complexity of the cells in the class $\mathcal{C}_m$ is $\mathrm{O}(n^2)$. This implies that the total complexity of the cells in an equivalence class is also $\mathrm{O}n^2)$ since these cells must all belong to $\mathcal{C}_m$ for some m.

Theorem 9: *The overall complexity of the cells in $\mathcal{C}_m$ is $\mathrm{O}(n^2)$.*

Proof: The longer proof of Theorem 3 will prove useful here. Recall that in that proof each vertex of the subdivision was given a label determined by the two rays on which it lies. The label consists of the names of the two emanating map vertices, as well as two bits specifying the layout of the anchor vertices with respect to these rays. There are four types of vertices corresponding to the four possible layouts. There could be as many as n vertices of Type 4 which carry the label of a particular pair of map vertices. It is the latter kind of vertices which can be large in number. Overall there are only $\mathrm{O}(n^2)$ vertices which are of Types 1-3 or lie on the boundary of the map polygon.

We now have to do a careful assignment of the edges and vertices of the subdivision to the visibility cell. The rays determining a vertex create four quadrants, one of which is the region where the two emanating vertices are blocked from view by the anchors. The vertex is assigned to the incident cell which lies in this quadrant. Similarly, each edge lies on a ray on one side of which the emanating vertex is hidden from view by the anchor vertex. The edge is assigned to the adjacent cell which lies on that side of the ray. Thus each edge/vertex is assigned to only one cell, but it still bounds all its neighboring cells.

This assignment of edges and vertices is motivated by our definition of visibility. Consider a ray emanating from v and anchored at w. From the view of any point p on this ray, the vertex v is not visible. Thus in the skeleton $V^*(p)$ there will be a spurious edge from w to v, and v will be considered a spurious vertex.

Consider now the edges which bound the cells in $\mathcal{C}_m$. Some of these edges could be portions of the boundary of the map polygon. There are at most nr rays in the subdivision and each ray will create one additional subdivision vertex on the boundary P. The number of subdivision edges lying on P cannot exceed the number of subdivision vertices on P, and these are at most $nr + n$ in number. We will ignore all such edges from now on since our goal is to prove a bound of $\mathrm{O}(n^2)$.

Consider any particular ray R emanating at the vertex v and anchored at the vertex w. This ray starts at w and proceeds away from v until it hits the boundary P. We will assume that the ray is directed away from v. There may be several subdivision vertices on the ray, besides its end-points, and this will divide R into a sequence of edges which bound some cells. We will think of these edges as open intervals separated by the vertices on R.

Assume that from the point of view of an observer sitting at v and looking in the direction of the ray, the anchor w lies to the left. We will argue only for the case of such "leftist" rays, the case of the rightist rays being similar. Each edge on the ray will be assigned to the cell lying to its left. However, we will have to count each such edge as contributing to the complexity of both the cell to its right as well as the one to its left.

As we trace along the ray from w to the boundary of P, we will traverse the edges on the ray in order. If the cell bounded by an edge (i.e. the cell to its right) is in $\mathcal{C}_m$, then the edge will contribute to the complexity of $\mathcal{C}_m$. Consider the vertices on this ray. Think of the ray as a vertical line which is directed towards the north. Each vertex x on this ray is caused by another ray, call it R', whose anchor may lie to the left or right of R and this doesn't really matter. The anchor for R' may lie below it – in this case the vertex is called an *incrementing* vertex. This means that as we cross v, a new vertex comes into view, and this vertex is the emanating vertex for R'. Similarly, a vertex is called *decrementing* if its anchor lies above R'. Upon crossing such a vertex, the emanating vertex falls out of view. The key observation is that a Type 4 vertex must be a decrementing vertex, and moreover its anchor must lie to the right of R.

Consider the sequence of edges we see that bound cells (to the right) which are in $\mathcal{C}_m$. Note that since all these edges are not assigned to the cells to the right, the cells will have v in their skeleton. Thus any two consecutive $\mathcal{C}_m$ edges will be separated by a collection of vertices (including their end-points) which contains an equal number of incrementing and decrementing vertices. We will label each $\mathcal{C}_m$ edge with the nearest incrementing vertex which lies below it. Since no two $\mathcal{C}_m$ edges are adjacent on R, each such edge gets assigned a unique label except possibly the first such edge on R. The crucial point here is that none of the labels can be a vertex of Type 4.

Each vertex can label at most four $\mathcal{C}_m$ edges since it lies on two rays and can be used as a label for at most two $\mathcal{C}_m$ edges on each of these two rays – one each when we are considering the cells to left or to the right of R. Since none of the labels is of Type 4, the number of distinct labels is $\mathrm{O}(n^2)$. Moreover, the number of unlabeled $\mathcal{C}_m$ edges is at most nr, i.e. one per ray. We conclude that the number of edges contributing to the complexity of cells in $\mathcal{C}_m$ is $\mathrm{O}(n^2)$.

It is easy to see that this bound can be reached in the example which proves the tightness of the $\mathrm{O}(n^3)$ bound on the complexity of the subdivision. $\square$

Consider any one embedding h of V^* and all the cells in the kernel of the embedded skeleton which belong to $\mathcal{E}_i$. Let $h(p)$ denote the point in P which has the same location relative to $h(v_o)$ as does p to v_o. The following theorem states that if $h(p)$ lies in one of the cells with skeleton V^* then $h(p)$ is a valid answer to the query $\mathcal{V}(p)$.

Theorem 10: *Let h be any embedding of $V^*(p)$ and $h(p)$ the corresponding location of p. Then, $\mathcal{V}(h(p)) = \mathcal{V}(p)$ if and only if $V^*(h(p)) = V^*(p)$.*

Proof: Clearly, if $h(p)$ and p have the same visibility polygon then they have the same skeleton. The non-trivial part is to show that if they have the same skeleton then they have the same visibility polygon.

Since the two points have the same skeleton, all the non-spurious vertices are identically laid out in the two visibility polygons, as are all the full edges. Consider now any fixed artificial edge e' of their skeleton. Suppose that e' is labeled as being in correspondence with a half edge. Then in the visibility polygon there is a spurious edge *se* which is adjacent to this half edge. The spurious edge starts at the same reflex vertex in both visibility polygons and is collinear with p. Thus the only difference between the two visibility polygons could be in the location of the spurious vertex where se meets the half edge. However, one end-point of the half edge is non-spurious and has the same location in both visibility polygons. Moreover, the line-characterization of the half edge is the same in both cases. Therefore, the location of the spurious vertex must also be identical in the two cases.

The other case to be considered is where the artificial edge e' is in correspondence with a partial edge. In this case there are two spurious edges in the visibility polygon which meet the partial edge. By an argument similar to the one above, it is easy to see that the location of the two spurious vertices must also be identical in the two visibility polygons.

We conclude that the relative location of all the spurious vertices must be identical in $\mathcal{V}(h(p))$ and $\mathcal{V}(p)$, and hence the two polygons must be identical. □

Thus, to verify if p could have been in any particular embedding, it suffices to check if $h(p)$ lies in a cell with the same visibility skeleton. Let k denote the total number of valid embeddings. Note that in general each embedding may contain multiple cells with the same visibility skeleton V^* as we are seeking. Our problem reduces to the following: given a collection of k families of disjoint convex polygons (one for each embedding) of total complexity e, we wish to identify the polygon(s) (if any) where a point q is located. When k is large, searching independently in each embedding's cells would require time at least k, which may be much larger than A (the output size). However, observe that all the embeddings are congruent and have the same location of p with respect to the origin. The only difference between the embeddings is in the visibility cells therein which have V^* as their skeleton. The cells in different embeddings are possibly totally unrelated to each other.

Our solution is to consider all embeddings at once by overlaying all their cells into one embedding of the skeleton. Thus, with reference to the origin of V^*, we have k collections of convex polygons of total complexity e_i that are overlaid to create a planar subdivision. The problem is now that of performing a point location in this subdivision: each region is labeled by the set of visibility cells that intersect to create it. The overall complexity and time to compute this subdivision is at most $\mathrm{O}(e_i^2)$.

Using data structures for point location in planar subdivisions due to Kirkpatrick [16] or Edelsbrunner, Guibas, and Stolfi [12], we can perform point location in time $\mathrm{O}(\log n)$, with preprocessing and space $\mathrm{O}(e_i^2)$. When we sum over all equivalence classes, our total cost will be

$$\mathrm{O}(\sum_{i=1}^{t} e_i^2) = \mathrm{O}(n^2 \sum_{i=1}^{t} e_i) = \mathrm{O}(n^2 N),$$

by Theorem 9. In most cases this will be the dominant preprocessing cost of our final algorithm.

There is actually one more problem to be overcome in implementing this approach: we need to associate with each region in these subdivisions we compute a list of the visibility cells whose intersection creates that region. This could blow up the space requirement by as much as another factor of n. To avoid this extra cost we proceed as follows. Consider a particular subdivision $\mathcal{S}$ obtained by the overlay of congruent visibility skeletons in class $\mathcal{E}_i$ and let $\mathcal{G}$ be the dual graph of that subdivision whose vertices are a set of canonical points, chosen one per region of $\mathcal{S}$. Now double each edge in $\mathcal{G}$ and consider an Eulerian tour τ of this new $\mathcal{G}$ (which must exist, as each vertex has even degree, and can be computed in linear time in the size of $\mathcal{G}$). Break the tour at some region covered by no cell (e.g. any infinite region of $\mathcal{S}$). The resulting path τ', can enter and exit each original cell multiple times, but its overall complexity will still be only e_i^2. Each passage of τ' through a cell can be thought of an interval. Let us construct an interval tree on this collection of e_i^2 known intervals. We can determine the cells covering any point of the path τ' by standard interval tree searching methods, improved by fractional cascading [6], in total time $\mathrm{O}(\log n + A)$, where A is the size of the answer. We can do the same for any point in $\mathcal{S}$, as its coverage will be the same as that for the canonical point selected in the region of $\mathcal{S}$ where it falls.

So we finally have our theorem:

Theorem 11: *The localization problem can be solved with preprocessing time and space of* $\mathrm{O}(n^2 N)$*, and a query time of* $\mathrm{O}(m + \log n + A)$.

Note that if we want any one solution, the query time drops to $\mathrm{O}(m + \log n)$.

Also, if we assume that P contains no collinear sides, then we can get a better preprocessing bound. Let equivalence class $\mathcal{E}_i$ contain s_i cells of total complexity e_i; each cell must come from a different embedding in this case. The cells are convex, and thus their overlay can have complexity at most $\mathrm{O}(s_i e_i)$ and can be computed within the same time bound [1]. Since each s_i is at most r, in this case the overall space and preprocessing time needed drops to $\mathrm{O}(rN)$.

We can also reduce the space in general, at the expense of increased query time. Moreover, we can smoothly trade-off the query time with the preprocessing time and space.

Theorem 12: *Let* $1 \leq f(n) \leq n$. *The localization problem can solved in* $\mathrm{O}\left(n^2 N / f(n)\right)$ *space and preprocessing, and query time* $\mathrm{O}(m + f(n) \log n + A)$.

Proof: Consider any particular equivalence class $\mathcal{E}_i$. Pick $\gamma = n^2/f(n)$ and consider the embeddings whose complexity is at most γ each. These can be partitioned into groups of embeddings such that each group has complexity roughly γ. The idea is to overlay the cells from the embeddings in a particular group, using a total space of $\mathrm{O}(\gamma^2)$ for each such group. The number of such groups cannot exceed $\mathrm{O}(n^2/\gamma)$ since the overall complexity of all the embeddings is bounded by $\mathrm{O}(n^2)$. The total space required by these groups is $\mathrm{O}(n^2\gamma)$. Searching independently in each group's planar subdivision requires time $\mathrm{O}(n^2 \log n/\gamma)$.

The embeddings of complexity at least γ each are searched independently also. Their total space requirement is $\mathrm{O}(n^2)$. Moreover, since they cannot be more than $\mathrm{O}(n^2/\gamma)$ in number, it requires $\mathrm{O}(n^2 \log n/\gamma)$ time to search these too. Thus, our total space requirement for this equivalence class is $\mathrm{O}(n^2\gamma)$ and the query time is $\mathrm{O}(n^2 \log n/\gamma)$. This implies the desired result. $\square$

The next result is obtained when we perform an independent point location in each distinct embedding.

Theorem 13: *The localization problem can be solved with* $\mathrm{O}(nN)$ *space and preprocessing and a query time of* $\mathrm{O}(m + r \log n + A)$.

6. Extensions and Variants

We briefly outline below how the above methods can be extended to work in a number of additional cases.

6.1. Map polygons with holes

When the map polygon has holes, the size and complexity of the visibility cell decomposition can be higher. In this case, we have a tight bound of $\Theta(n^2r^2)$ on the number of visibility cells. Althouh we do not have the space to develop this here, we just mention that when all the holes are convex, the increase in the number of visibility cells can be bounded in terms of the *number of holes*. This also applies to the increase in the preprocessing and space bounds.

6.2. The limited range version

We now consider a feature of range-finders that arises in practice — they can reliably obtain range readings only up to some distance D [19]. Beyond this distance, the noise levels are too high to measure the distance, and we only learn that the distance is greater than D. Our approach to preprocessing $\mathcal{P}$ can be modified to work even in this case. The set of points within distance D of an edge of $\mathcal{P}$ is an oval region. Consider now the arrangement of the oval regions defined by all the edges of $\mathcal{P}$; since any two ovals intersect at most at four places, this arrangement partitions the plane into $\mathrm{O}(n^2)$ subdivisions. Within each cell of this subdivision, the set of edges of $\mathcal{P}$ that are within distance D is invariant. Intersecting this arrangement with our visibility cell decomposition, we obtain a modified decomposition with the property that in each subdivision, the (redefined) skeletons are the same. Our search process is now applied to this modified decomposition to obtain the same query time.

6.3. The single-shot query problem

Consider now the problem of answering a single query $\mathcal{V}$. Here the cost of any preprocessing must be included in the cost of answering the query. We present an algorithm running in time $\mathrm{O}(nm)$ based on some results in *ray-shooting*. Suppose we wish to determine, at each vertex of $\mathcal{P}$, where the ray going in a fixed direction first hits $\mathcal{P}$ again. This problem is equivalent to trapezoidalizing $\mathcal{P}$ using lines parallel to the shooting direction. This can be done in linear-time [5, 13] given a triangulation of the polygon $\mathcal{P}$. The recent result of Chazelle [4] shows that the triangulation itself can be computed in linear time.

Theorem 14: *Given a map polygon $\mathcal{P}$ and a visibility polygon $\mathcal{V}$, the set of valid locations of p can be determined in time $\mathrm{O}(nm)$.*

Proof: First, for each of the $\mathrm{O}(m)$ spurious edges in $\mathcal{V}$ we determine the trapezoidalization of $\mathcal{P}$ in the direction of that spurious edge. This requires a total of $\mathrm{O}(nm)$ time. Now, we try each possible embedding of $\mathcal{V}$ in $\mathcal{P}$, of which there are at most n. In each potential embedding case, we can determine its validity by using the information from the trapezoidalizations queries. Effectively, we are trying to trace a fixed placement of $\mathcal{V}$ in $\mathcal{P}$; we do this by following edges of P and using the trapezoidlizations to implement in constant time ray-shooting queries when we need to walk along a spurious edge of $\mathcal{V}$. Thus the time to verify an embedding is $\mathrm{O}(m)$ and the total time is $\mathrm{O}(nm)$. □

When the range-finder has a limited range D, we can answer a single query in $\mathrm{O}(n^2)$ time.

6.4. Visibility query processing

Consider the problem of preprocessing a polygon $\mathcal{P}$ so that a visibility query can be efficiently answered. A visibility query is a point $p \in \mathcal{P}$, and we are required to compute the edges and vertices of P visible from p. An interesting side effect of our results is the construction of an efficient data structure for this problem. The obvious approach is to compute the visibility cell decomposition and store in each cell a modified skeleton for the cell. This new skeleton records the (circular) list of the visible edges and vertices for that cell — which is constant over the cell. A query can now be answered by performing a point location in this cell decomposition. However, this is inefficient in its use of space since we must store the skeletons for each cell separately.

We now sketch an idea for making the space requirement linear in the complexity of the cell decomposition. The idea is to avoid storing the full visibility list at each cell. Instead, at each interior line we store the change in the visibility as we cross it. Borrowing on an idea of Chazelle [3], we actually store the visibility skeleton only at those cells where the visibility is a "local" minimum (i.e., those cells that see less of P than any of their neighbors). These are the cells where crossing any boundary edge leads to additional vertices/edges becoming visible. To compute the visibility in any cell, we perform a walk to a minimal visibility cell, keeping track of the changes in visibility as we cross interior lines. If we are not at a minimal cell, there is always a cell boundary such that crossing it will cause the number of visible vertices to decrease. The length of the walk cannot exceed the size of the output.

Furthermore, the number of these minimal visibility cells is $\mathrm{O}(n^2)$. This is because there cannot be two adjacent Type 4 vertices on the boundary of such a minimal cell. One way to see this is to use the analysis in the proof of Theorem 9, except that we apply it to edges which bound a minimal visibility cell, rather than to edges bounding cells in $\mathcal{C}_m$ as was the goal in that Theorem. In particular, consider traversing a ray away from the anchor which (say) lies on the left of the ray; then, a minimal cell bounded by an edge e lying on this ray must also lie to the left of the ray. Further, the first endpoint of e that is encountered must be a decrementing vertex (and possibly of Type 4), but the second endpoint of e must be incrementing and hence cannot be of Type 4. It follows that each minimal cell has at least one vertex not of Type 4, and we chanrge the minimal cell to this vertex. In this fashion each vertex can be charged at most four times. By recalling that the number of vertices that are not of Type 4 is $\mathrm{O}(n^2)$, we get:

Theorem 15: *Using $\mathrm{O}(n^3)$ space and preprocessing, a visibility query can be answered in time $\mathrm{O}(\log n + m)$ where m is the size of the output visibility polygon.*

7. Further Work

- The single-shot problem resembles a classic string-matching problem, and it is likely that an algorithm running in time $o(mn)$ can be devised using techniques from that field.

- We assumed that the robot has a compass and thus $\mathcal{V}$ is oriented with respect to $\mathcal{P}$. What if this assumption were removed?

- In any real-life situation the data obtained friom the sensor is likely to be noisy. How can se solve this problem when we seek only approximate congruence between $\mathcal{V}$ and the visibility polygon of a point in $\mathcal{P}$?

- A hard but natural extension of our problem is to the case of 3-dimensional polyhedral terrains. Here the robot's viewing mechanism would be a camera which would deliver two-dimensional images.

Acknowledgements

We gratefully acknowledge useful discussions with Bernard Chazelle, John Hershberger, Simeon Ntafos, and Emo Welzl.

References

[1] B. Aronov, M. Bern, and D. Eppstein. Arrangements of Polytopes. Unpublished manuscript, 1991.

[2] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. In *Proceedings of the 4th Canad. Conf. Comput. Geom.*, pages 23–28, 1992.

[3] B. Chazelle. An improved algorithm for the fixed-radius neighbor problem. *Inform. Process. Lett.*, 16:193–198, 1983.

[4] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6:485–524, 1991.

[5] B. Chazelle and J. Incerpi. Triangulation and shape-complexity. *ACM Trans. Graph.*, 3(2):135–152, 1984.

[6] B. Chazelle and L. Guibas. "Fractional cascading: I. A data structuring technique. *Algorithmica*, 1:133–162, 1986.

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

[8] I.J. Cox. Blanche — an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. Robotics and Automation*, 7:193–204, April 1991.

[9] I.J. Cox and J.B. Kruskal. On the congruence of noisy images to line segment models. In *2nd International Conference on Computer Vision*, pages 252–258, 1988.

[10] I.J. Cox and J.B. Kruskal. Determining the 2- or 3-dimensional similarity transformation between a point set and a model made of lines and arcs. In *Proceedings 28th IEEE Conference on Decision and Control*, pages 1167–1172, 1989.

[11] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[12] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15:317–340, 1986.

[13] A. Fournier and D. Y. Montuno. Triangulating simple polygons and equivalent problems. *ACM Trans. Graph.*, 3(2):153–174, 1984.

[14] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear time algorithms for visibility and shortest path problems inside simple polygons. In *Proceedings of the 2nd Annu. ACM Sympos. Comput. Geom.*, pages 1–13, 1986.

[15] L. Guibas, M. Overmars, and M. Sharir. Intersecting line segments, ray shooting, and other applications of geometric partitioning techniques. In *Proceedings of the 1st Scand. Workshop Algorithm Theory*, volume 318 of *Lecture Notes in Computer Science*, pages 64–73. Springer-Verlag, 1988.

[16] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12:28–35, 1983.

[17] K. Mehlhorn. *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry.* Springer-Verlag, 1984.

[18] D.P. Miller, D.J. Atkinson, B.H. Wilcox, and A.H. Mishkin. Autonomous navigation and control of a Mars rover. In *Automatic Control in Aerospace. Selected papers from the IFAC Symposium, Tsukuba, Japan, 1989*, pages 111–114. Pergamon, Oxford, UK, 1989.

[19] G.L. Miller and E.R. Wagner. An optical rangefinder for autonomous robot cart navigation. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehicles.* Springer-Verlag, Berlin, 1990.

[20] C.N. Shen and G. Nagy. Autonomous navigation to provide long distance surface traverses for Mars rover sample return mission. In *IEEE International Symposium on Intelligent Control*, pages 362–367, 1989.

[21] S.S. Skiena. *Geometric Probing.* PhD thesis, University of Illinois, Urbana-Champaign, April 1988. Also available as Computer Science Dept. Report UIUCDCS-R-88-1425.

[22] C. Ming Wang. Location estimation and uncertainty analysis for mobile robots. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehicles.* Springer-Verlag, Berlin, 1990.