Simple Robots with Minimal Sensing: From Local Visibility to Global Geometry

Subhash Suri[*]

Department of Computer Science
University of California
Santa Barbara, USA 95106
suri@cs.ucsb.edu

Elias Vicari[†] and **Peter Widmayer**[†]

Institute of Theoretical Computer Science
ETH Zurich
8092 Zurich, Switzerland
{vicariel,widmayer}@inf.ethz.ch

Abstract

We consider problems of geometric exploration and self-deployment for simple robots that can only sense the combinatorial (non-metric) features of their surroundings. Even with such a limited sensing, we show that robots can achieve complex geometric reasoning and perform many non-trivial tasks. Specifically, we show that one robot equipped with a single pebble can decide whether the workspace environment is a simply-connected polygon and, if not, it can also count the number of holes in the environment. Highlighting the subtleties of our sensing model, we show that a robot can decide whether the environment is a convex polygon, yet it cannot resolve whether a particular vertex is convex. Finally, we show that using such local and minimal sensing, a robot can compute a proper triangulation of a polygon, and that the triangulation algorithm can be implemented collaboratively by a group of m such robots, each with $\Theta(n/m)$ memory. As a corollary of the triangulation algorithm, we derive a *distributed* analog of the well-known Art Gallery Theorem: a group of $\lfloor n/3 \rfloor$ (bounded memory) robots in our minimal sensing model can self-deploy to achieve visibility coverage of an n-vertex art gallery (polygon). This resolves an open question raised recently by Ganguli et al.

Introduction

The study of simple robot systems, with minimal sensory input, is of fundamental interest in both theory and practice. In theory, a minimalistic model provides a clean conceptual framework for performance analysis and lends insights into the inherent complexity of various tasks: the positive results identify the easy problems, while the negative results help isolate the difficult problems that necessitate richer functionality and sensing. Models with simple sensing are also robust against noise and help simplify the information (belief) space of the robot systems, whose complexity is often a source of much difficulty in planning (LaValle 2006). On the practical side as well, robots with a simple sensing architecture have many advantages: they are inexpensive, less susceptible to failure, robust against sensing uncertainty, and useful for different applications. With the emergence of *wireless sensor networks* (Pottie & Kaiser 2000), a group of simple micro-robots also offers an attractive and scalable architecture for large-scale collaborative exploration of unknown environments.

Motivated by these considerations, we investigate several fundamental computational geometry problems in the context of simple robots, with minimal sensing ability. In particular, we assume that the robot is a (moving) point, equipped with a simple camera that allows the robot to sense just the *combinatorial features* in its field—these features are *unlabeled*, however, meaning that the sensor can detect a vertex of the polygon, but all vertices look the same to it. The features are represented as a binary vector, called the *combinatorial visibility vector* and denoted cvv. This vector is defined by the *cyclically ordered* list of vertices visible from the robot's current position, where each bit indicates whether or not the consecutive vertices have a polygon edge between. When this bit is 0, meaning that the consecutive vertices do not form an edge, some portion of the polygon is invisible from the current location of the robot. See Figure 2 for an example of a combinatorial visibility vector.

We assume that the visible vertices are always presented in a *consistent* cyclic order. Without loss of generality, and to be concrete, we may assume that the order is counterclockwise around the robot's position. In particular, for the robot located on the polygon boundary, the first vertex in the combinatorial visibility vector is a neighbor of v; for the robot located in the interior, we make no assumption about the identity of the first vertex—it can be arbitrarily chosen. It is worth pointing out, however, that this is a *local* property, and does not assume an ability to distinguish between *global* clockwise and counterclockwise traversals of boundary components. Figure 1 illustrates this distinction: the cyclic visibility order at p is identical in both figures, yet following the boundary in the direction of the *first* edge leads to different global cyclic orientations—clockwise in the left figure, and counterclockwise in the right figure. The robot has no other sensing capability, and thus receives no information about distances, angles, or world coordinates—

[*]Work done while the author was a visiting professor at the Institute of Theoretical Computer Science, ETH, Zurich. The author wishes to acknowledge the support provided by the National Science Foundation under grants CNS-0626954 and CCF-0514738.

[†]The authors are partially supported by the National Competence Center in Research on Mobile Information and Communication Systems NCCR-MICS, a center supported by the Swiss National Science Foundation under grant number $5005 - 67322$.

in particular, for three consecutive points p, q, r along its path, it cannot determine whether $\langle p, q, r\rangle$ is a right turn, or a left turn, namely, whether r lies to the left or to the right of the directed line defined by p and q. We assume idealized abstract sensing throughout, since our primary goal is to understand theoretical limits of our model. Our algorithms are deterministic and analyzed in the worst-case model.

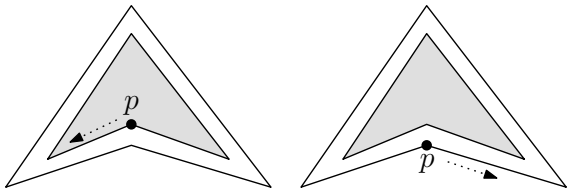

Figure 1: The robots cannot distinguish a global cyclic orientation.

We are particularly interested in tasks that are seemingly *global*. For instance, can one or more robots decide whether a (unknown) polygonal environment (the robots' workspace) is *simply connected*, or it has holes? We show that a single robot with a single pebble (marker) is able to decide if the workspace is simply connected; equivalently, a secondary passive robot can play the role of the pebble. Generalizing this, we show that if the workspace is a polygonal environment with k holes, then a single robot with one pebble can detect all the holes.

Interestingly, in our minimalistic model of sensing, there are simple topological questions that seem undecidable. For instance, while it is quite easy for a single robot to decide whether the workspace is a convex polygon, it is not possible to decide whether a *particular vertex* is convex or reflex! Similarly, a robot cannot decide which is the *outer boundary* of the multiply connected polygonal workspace, although it can discover and count all the boundary components in the environment.

Finally, we show that using such local and minimal sensing, a robot can compute a proper triangulation of a polygon. Furthermore, our triangulation algorithm can be implemented distributively by a group of m robots, each with $\Theta(n/m)$ words of memory, where we make the standard assumption that each word has $\Theta(\log n)$ bits. As a corollary of the triangulation algorithm, we derive a distributed analog of the well-known Art Gallery Theorem (Chvátal 1975): a group of $\lfloor n/3\rfloor$ robots can self-deploy to guard a simple polygon. This improves the recent result of (Ganguli, Cortes, & Bullo 2006), where they showed that $\lfloor n/2\rfloor$ distributed guards can achieve the coverage, raising the tantalizing question whether the distributed nature of the problem is the source of the gap between the centralized optimum of $\lfloor n/3\rfloor$ and their bound. Our result shows that even with minimal sensing and distributed robots, $\lfloor n/3\rfloor$ guards suffice. Indeed, triangulation is a fundamental data structure, used as a building block of many geometric algorithms, and so we expect that this basic result will find other applications as well.

Related Work

Combinatorial geometric reasoning is key to many motion planning and exploration tasks in robotics (Latombe 1991; LaValle 2006). Our work is similar in spirit to that of (Tovar, Freda, & LaValle 2007; Yershova *et al.* 2005), in that we aim to explore the power and limitations of a minimal model of robot systems. Our model is different from the one in (Tovar, Freda, & LaValle 2007) because they assume that important features of the environment are uniquely *labeled*, allowing sensors to distinguish these *landmarks*. Our model is similar to the one used in (Yershova *et al.* 2005) but the nature of problems investigated in our paper is quite different from those studied in (Guibas *et al.* 1999; Sachs, Rajko, & LaValle 2004; Yershova *et al.* 2005): their main focus is navigation and pursuit evasion, while we are concerned with the geometric and topological structure of the environment and collaborative and distributed self-deployment, which do not seem to have been addressed in the past.

A Minimal Sensing Model

We consider robot systems with a simple model of "visual" sensing. At any position p, the sensory input of the robot is a *combinatorial visibility vector* $\mathrm{cvv}(p)$, which is a cyclically ordered vector of zeroes and ones. This vector is defined by the vertices of the polygonal environment that are visible from the point p, and the binary bits encode whether or not the consecutive vertices form an edge of the polygon (see Figure 2). As mentioned earlier, we assume the visible vertices are always given in a consistent, say, counterclockwise, order around the robot's position. Thus, the visibility vector tells the robot the number of vertices visible to it, and a cyclic order of the *edge types* (boundary or diagonal) defined by consecutive vertices. Figure 2 shows an example, with the combinatorial visibility vector of vertex p.

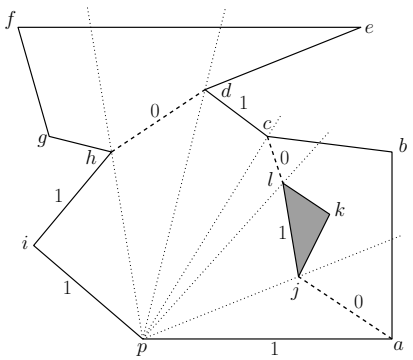

Figure 2: Illustration of the combinatorial visibility vector. In cyclic (counterclockwise) order, the vertices visible from p are p, a, j, l, c, d, h, i, and its visibility vector is $\mathrm{cvv}(p) = (1, 0, 1, 0, 1, 0, 1, 1)$.

We emphasize that the polygon induced by the visibility vector is different from the visibility polygon—the former only uses the vertices of the original polygon, and is typically a strict subset of the visibility polygon. We believe

that the visibility vector, despite being less informative than the corresponding visibility polygon, is better suited for our simple sensing model: in our coordinate-free, binary sensing model, there is no obvious way to represent or communicate entities other than polygon vertices or edges. Even polygon vertices and edges have no "names" or labels, and as such they can only be described in relative terms: e.g., ith vertex in the cyclic order of the visibility vector of vertex q. In this regard, our sensing model is more basic and weaker than the minimalism assumed by (Tovar, Freda, & LaValle 2007), who require presence of labeled features and distinguishable landmarks in the environment.

In our algorithms, we utilize a *pebble* as a marking device to make a particular vertex distinguishable to the robot; this is a standard tool used also in (Yershova *et al.* 2005). Another robot can also play this (passive) role. In our distributed setting with multiple robots, we assume that robots are visually indistinguishable but possess distinct identifiers, which can be learnt through communication. The combinatorial visibility vector is appropriately enhanced to indicate the vertices that have at least one robot located at them.

In terms of the robot's motion abilities, we only assume that (1) it can sense when it is on a boundary, (2) it can move cyclically along a boundary, and (3) at a position p, it can choose to move towards a vertex visible from position p. For the workspace, we assume a polygonal environment, which may be simply or multiply-connected (with holes), with an unknown geometry and an unknown number of vertices.

Convexity of the Environment

We begin with a simple example highlighting the severe limitations of our sensing model: a robot in our model cannot decide whether a particular vertex of the environment is convex or reflex. Consider the symmetric polygon shown in Figure 3: the visibility vectors of the pairs of vertices $\{a, c\}$ and $\{b, d\}$ are identical: $\mathrm{cvv}(a) = \mathrm{cvv}(c) = (1, 1, 1, 1)$, while $\mathrm{cvv}(b) = \mathrm{cvv}(d) = (1, 0, 1)$. The key observation is that a is a reflex and c a convex vertex, yet their visibility vectors (and those of their neighbors) are indistinguishable. Thus, the convex or reflex type of a vertex cannot be decided by simply looking at the visibility vectors of the polygon vertices. Moreover, if we choose the "width" of this corridor-shaped polygon to be arbitrarily small, then the robot also cannot gain any other visibility information from the visibility vectors computed at points other than the vertices. Indeed, if the robot moves ε away from the vertex a and the width of the polygon is sufficiently smaller than ε, then its visibility vector immediately switches to vectors of one of its neighbors. Thus, using only the visibility vector information available to our robot, it cannot distinguish between the vertex types (convex or reflex) in our model.

This may seem surprising in light of the fact that such a robot *can* decide whether the entire polygon is convex or not. Observe that a necessary condition for the polygon to be convex is that the visibility vector of any vertex v must be all 1's—the presence of a 0 bit implies that there exists a "pocket" in the polygon not seen by v. If the visibility vectors of *all vertices* are all 1's, then the polygon is convex—this follows because every non-convex polygon must contain

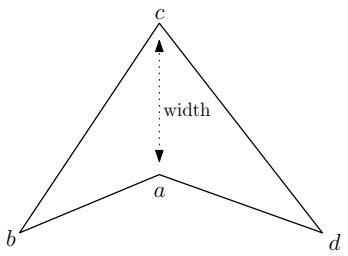

Figure 3: An example showing that a robot in our model cannot resolve whether a vertex is convex or not. The sensing information (combinatorial visibility vectors) is the same for a and c, yet they have different types (convex, reflex).

a reflex vertex, and if r is reflex vertex them the visibility vector of the (either) neighbor of r cannot be all 1's. Thus, an algorithm to decide the convexity of a polygon P is the following.

> Start at any vertex, say, v_0, and consider the visibility vector $\mathrm{cvv}(v_0)$. If this vector is not all 1's, stop—the polygon is ostensibly non-convex. Otherwise, compute the size of the polygon, say, n by counting the number of 1's in the vector. Repeat the convexity test from each of the remaining $n - 1$ vertices, by moving cyclically along the boundary. If the visibility vectors of all the vertices are all 1's, then the polygon is convex.

We, thus, have the following theorem.

Theorem 1 *A single robot in our binary sensing model can decide whether a given polygon (workspace) is convex or not. However, it is not possible to decide if a particular vertex is convex.*

Topological Structure of the Environment

Next, we consider another fundamental task related to the geometry of the environment: is the workspace of the robot simply-connected, or does it have holes? If the polygon is multiply-connected, then how many holes does it have? We first show that one robot, equipped with a single pebble, can decide the simplicity question. The pebble, of course, can be replaced by another (passive) robot. Subsequently, we show that with at least one pebble (or passive robot), the robot can also identify and count the number of topological holes in the workspace.

Is the polygon simply-connected?

In order to decide the topological simplicity of its environment, the robot has to determine whether the 1-edges in its combinatorial visibility vector belong to two (or more) different component boundaries. As an example, how does the robot decide when located at vertex p in Figure 2, whether the edge (l, j) is part of a "hole" or does the outer boundary connect to it in the back (near vertex k, which is invisible to p)? We begin with a simple geometric fact that plays an important role in our algorithm.

Lemma 1 *Suppose P is a multiply-connected polygon, and C is a cycle representing one of the components. Then, there*

always exists a vertex $u \in C$ that sees a vertex $v \in P \setminus C$. That is, there must be a vertex $u \in C$ whose visibility vector includes a vertex from a different component of P.

Proof: The proof follows from the fact that any polygonal domain admits a triangulation (using only diagonals that connect vertices). Such a triangulation is a connected graph, and so the boundary C of any component includes at least one diagonal in the triangulation that connects it to another component. ∎

We utilize this lemma to decide simplicity of the workspace as follows: the robot moves to a boundary, and traverses it in cyclic order; at every vertex, it checks each vertex of its visibility vector to see if it lies on the same boundary or not. By the preceding lemma, if the polygon has a hole, then one of these tests will necessarily fail. The following pseudo-code describes our algorithm. The step that checks whether a vertex v is on the same boundary as vertex u is done through a function EXPLORE, which is described right after the main algorithm.

SIMPLICITY (P)

1. Initially, the robot moves to an arbitrary vertex of the boundary.
2. The robot marks that vertex with a pebble, and walks along the boundary (finishing when it returns to the pebble), to determine the number of edges on this boundary. Let this number be n_0.
3. The robot now begins the main phase of the algorithm. The vertex with the pebble currently on it represents the vertex being scanned. Let u_0 be the initial vertex with the pebble. Repeat the following steps n_0 times.
   (a) Let u_i be the vertex with the pebble on it, and let $\text{CVV}(u_i)$ be the visibility vector of u_i. Perform EXPLORE (u_i, v_j, n_0), for each vertex v_j visible from u_i. (The robot moves to vertex v_j, performs the EXPLORE operation, and then returns to u_i.)
   (b) If any call to EXPLORE returns $simple = false$, then terminate; the polygon is multiply-connected. Otherwise, once all the vertices visible from u_i have been explored, advance the pebble from u_i to u_{i+1}.

EXPLORE (u, v, n)

1. Starting at v, cyclically move along the component boundary containing v.
2. If a pebble is encountered within n steps, set $simple = true$; and stop (the robot is back at vertex u_i).
3. Otherwise, set $simple = false$; retrace n steps backwards (returning to vertex v), and move to the vertex u (identified with a pebble).

For instance, in Figure 2, the call EXPLORE $(p, a, 10)$ returns true, while EXPLORE $(p, j, 10)$ would return false. We summarize this result in the following theorem.

Theorem 2 *A single robot with a pebble can decide if a polygonal environment is simply connected or not.*

The processing time of the algorithm can be improved in several ways, though that's not our main intent here. For instance, instead of checking all vertices of the combinatorial visibility vector, it suffices to limit the EXPLORE to only the endpoint of the 0-edges.

The use of a pebble to mark a vertex in our algorithm seems critical. While we do not attempt to formalize this as an impossibility result, the following example (see Figure 4) suggests some of the difficulties. Suppose that an algorithm $\mathcal{A}$ exists that correctly decides with t movements that the polygon on the left in Figure 4 is non-simple. Then running this algorithm from a vertex in the middle of the polygon depicted on the right (whose size is proportional to t), the algorithm has to conclude that this polygon is also is non-simple, since at every step the two polygon yield identical visibility information. This argument is not completely satisfying because the size of the polygon depends on the running time of the algorithm, however, it does suggest the difficulty in designing such an algorithm.

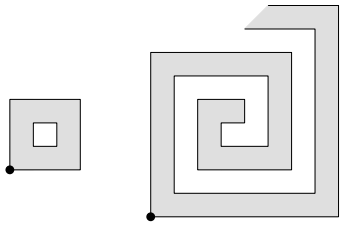

Figure 4: A robot without pebbles cannot decide simplicity.

We next show how a robot can discover all the components (holes) in its workspace environment.

Counting the number of holes

The algorithm is a recursive extension of the simplicity algorithm: when the robot discovers a new hole, it recursively scans its boundary to discover new holes. Pebbles are used to mark the holes that have been already discovered. The key difference from the simplicity algorithm is that the robot needs to maintain state information to go back in the recursion. The following pseudo-code describes the algorithm.

COMPONENTS (P)

1. The robot starts on the boundary of an arbitrary component. It marks this component with one pebble, computes its size (number of edges), by walking around the boundary, using a pebble as a marker. Let n_0 be the size of the current component.
2. For the component being scanned, the robot maintains the index (in cyclic order, from the starting vertex) of the current vertex. Let u_i be the current vertex. For each vertex $v_j \in \text{CVV}(u_i)$, in cyclic order, the robot invokes EXPLORE(u_i, v_j, n_0).
3. If EXPLORE(u_i, v_j, n_0) returns true, then scan of the current component continues.
4. If EXPLORE(u_i, v_j, n_0) returns false, then we have discovered a new component.

(a) The robot then uses a new pebble to mark this component; increments the component counter; computes the size, n', of the new component, and sets $n_0 = \max\{n_0, n'\}$, as the maximum length of the walk in EXPLORE.

(b) The robot saves state for the old component (containing the vertex u_i), along with how to return to u_i from v_j (using pebbles),[1] and then recursively works on the new component.

Theorem 3 *If the polygonal environment consists of k holes, then a robot with $k+1$ pebbles (one singly distinguishable) can discover all the components.*

For the sake of simplicity, we have described our algorithm using $k+1$ pebbles. The number of pebbles can be reduced to 1, albeit at the expense of increasing the total amount of movement incurred by the robot(s). Due to lack of space, we omit the details of this extension, and simply state the main theorem.

Theorem 4 *A robot with a single pebble can discover all the boundary components of a (multiply-connected) polygonal environment.*

In our algorithm, we assumed that the robot has $\mathcal{O}(k)$ memory, where k is the number of components, to save the state of the recursion. Alternatively, the same result can be obtained with $\mathcal{O}(k)$ robots, each with constant (word) memory.

Which is the outer boundary?

Finally, as another example of a simple task that is difficult in our minimal model, we argue that even though a robot can discover all the components (holes) in the environment, it cannot decide *which* one is the outermost boundary. Consider the centrally symmetric polygon shown in Figure 5. We observe that every vertex in this polygon has the same combinatorial visibility vectors, namely, $(1, 0, 1, 1, 0, 1)$, irrespective of whether it is on the outer cycle or the inner cycle. Because the robot cannot sense or measure angles or distances, its world view looks the same whether it is on the outer cycle or the inner one, implying that it cannot resolve between those two boundaries.

In the next section, we show a single robot, using a constant number of pebbles, is able to compute a proper triangulation of a simple polygon. This result, in addition to providing a important data structure for geometric reasoning and exploration, is also the basis for solving the art gallery problem in our sensing model.

[1] Observe that the robot is able to identify v_j from the visibility vector of u_i, but not vice versa. In order to identify u_i from v_j, the robot must place a "distinct" pebble at u_i, move to v_j, and then record the index of u_i in the $\mathrm{cvv}(v_j)$. This pebble must be distinct in the sense that robot can distinguish it from any other pebbles that are marking other components. One can achieve this in several way: by using two pebbles, instead of one, if the robot is able to make this distinction; or using a special pebble that is used for this signaling.

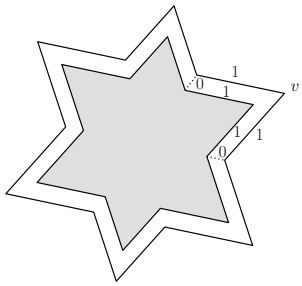

Figure 5: An example showing that a robot in our model cannot resolve which is the outer boundary.

Polygon Triangulation

In this section, we describe our algorithm for triangulating a simple polygon, which is the key step in solving the art gallery problem as well as a geometric structure with broad applicability (de Berg *et al.* 1997). We describe our algorithm for a single robot with a constant number of pebbles in the minimal model, with the assumption that this robot has $\Theta(n)$ memory to store the triangulation. Later we show that the same result can be obtained by a collaborative group of m robots, each with $\Theta(n/m)$ memory. As a corollary of this distributed triangulation, we obtain that a group of $\lfloor n/3 \rfloor$ robots, each with $\Theta(1)$ memory, can collectively build the triangulation and solve the art gallery problem.

Our polygon triangulation algorithm is recursive in nature, and works as follows. See Figure 6. The robot places an initial pebble at a vertex, call it v_0. The combinatorial visibility vector of v_0, $\mathrm{cvv}(v_0)$, consists of a sequence of 0-edges intermixed with 1-edges. Each 0-edge is a diagonal that separates a "pocket" from v_0, and the pockets defined by different 0-edges are pairwise disjoint. The robot will *recursively* triangulate the pockets formed by the 0-edges (say, by visiting them in cyclic order), and then complete the triangulation by drawing diagonals from v_0 to all the vertices in its visibility vector. Thus, in high-level pseudo-code, the triangulation algorithm can be described as follows:

TRIANGULATION (P)

1. The robot begins at an arbitrary vertex v_0. Let $e_1, e_2, \ldots, e_k$ denote the 0-edges in the combinatorial visibility vector of v_0 (in ccw order), and let P_i denote the pocket of the polygon defined by the edge e_i.
2. The robot recursively computes the triangulation of P_i, for $i = 1, 2, \ldots, k$.
3. The robot finishes the triangulation by adding diagonals from v_0 to all the vertices in its combinatorial visibility vector (endpoints of both 0 and 1-edges).

Turning this high level description into a correct algorithm that fits in our minimal sensing model, however, requires several careful steps. We use the illustration of Figure 6 to explain the key steps of the algorithm.

At the top level of the recursion, the base visibility vector is defined for vertex v_0. The robot consistently scans the

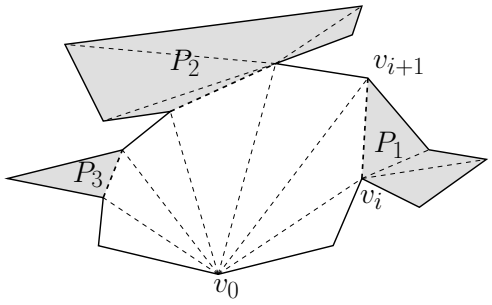

Figure 6: The triangulation algorithm. Starting at v_0, the algorithm triangulates the pockets P_1, P_2, P_3, in that order, and finally completes the triangulation by drawing edges from v_0 to all vertices visible from it.

edges of a visibility vector in cyclic order. Let the first 0-edge in this visibility vector be v_iv_{i+1}, and let P_i denote the pocket (subpolygon) defined by this edge. The vertices v_i and v_{i+1} are *identified* by the robot as the ith and the $(i+1)$st vertices in $\mathrm{cvv}(v_0)$. However, as the robot enters the pocket P_i for recursive triangulation, it no longer "sees" the polygon from v_0, and needs a way to distinguish (identify) v_{i+1} from the "new base" v_i. Similarly, the robot needs a way to recognize v_0 from v_i, where it must return after the recursion ends in the pocket P_i. The robot does this using two pebbles as follows. (The location of v_0 can also be stored by counting the number of boundary edges in the cyclic ordering. But, for the sake of uniformity, we use the pebbles to transfer this state information.)

Lemma 2 *Using 2 pebbles, the robot can compute two indices j_1 and j_2 such that v_{i+1} and v_0, respectively, are the j_1th and j_2th vertices in the cyclic ordering of the combinatorial visibility vector* $\mathrm{cvv}(v_i)$.

Proof: From v_0, which is marked by the first pebble, the robot heads straight towards v_{i+1}, drops a second pebble there, and returns to v_0. It then, heads to v_i, and identifies v_{i+1} as the first vertex in the cyclic order of $\mathrm{cvv}(v_i)$ that has a pebble—its index is the value j_1. Similarly, the index of the second vertex with a pebble serves as the value j_2. Having identified both v_0 and v_{i+1} in the local visibility of v_i, the robot can then recover these pebbles, ensuring that we only use at most two pebbles throughout the algorithm. ∎

Note that the procedure mentioned in the proof of the previous Lemma can be easily simulated by a robot with only one pebble, at the cost of an increased amount of total movements. Once the robot can identify v_{i+1} from its local view, it knows the extent of the pocket P_i: the edge v_iv_{i+1} marks the end of the pocket and the recursive call to the triangulation.

Secondly, by always visiting the pockets in a cyclic order, the robot can consistently compute a "vertex labeling" that serves to identify and store the triangulation globally. In particular, while at position v_0, the robot can see all the vertices in its visibility vector, that "view" is entirely local—the jth ccw vertex in $\mathrm{cvv}(v_0)$ has no meaning to the robot when it is located at another vertex v_k. Thus, during the triangulation algorithm, the robot computes a global labeling, which is a cyclic ordering of the vertices in P, starting from the base vertex v_0. This labeling is computed easily as follows. The robot assigns the label 0 to the vertex v_0; that is, $\ell(v_0) = 0$. It then assigns increasing labels to all the vertices in $\mathrm{cvv}(v_0)$ until it comes to the first 0-edge, say, $e_i = (v_i, v_{i+1})$. As the robot recursively computes the triangulation of P_i, it assigned labels in the pocket, starting with $\ell(v_i)$, and ending with the label $\ell(v_{i+1})$. At this point, the robot continues the labeling in $\mathrm{cvv}(v_0)$ until the next 0-edge, and so on. In the end, the triangulation is stored in the robot's memory as a collection of diagonals, where diagonal (i, j) means the presence of a triangulation edge between the vertices are have indices i and j in the ccw walk along the polygon starting at v_0.

So far we have described the triangulation algorithm using a single robot with $\Theta(n)$ memory (necessary to store the triangulation). But it is easy to convert this into a distributed implementation, using a group of m robots, each with $\Theta(n/m)$ memory. In the distributed model, we only assume that each robot has a unique ID and that robots co-located at a common vertex can communicate (a rather weak communication requirement). Then, the single-robot algorithm can be simulated easily in this new setting: one robot acts as a leader and execute the triangulation algorithm, while the others follow passively the leader, acting as a storage device and simulating the role of pebbles, when required. This is possible, because the target vertex of the leader's motion can be specified by an index of the cvv and transmitted to every robot. This shows that $\lfloor n/3 \rfloor$ robots, each with $\mathcal{O}(1)$ memory, can achieve the triangulation.

Theorem 5 *In our minimal sensing model, one robot with a pebble can compute a triangulation of a simple polygon. The algorithm is easily turned into a distributed implementation using m robots, each with $\Theta(n/m)$ memory.*

The Art Gallery Problem

The well-known Art Gallery Theorem (Chvátal 1975; Fisk 1978) asserts that every n-vertex polygon can be "guarded" by placing $\lfloor n/3 \rfloor$ guards at vertices of the polygon, and this bound is the best possible in the worst-case. The classical setting of the art gallery theorem assumes full knowledge of the polygons, including vertex coordinates. In a recent paper, (Ganguli, Cortes, & Bullo 2006) showed that, given an unknown polygon, $\lfloor n/2 \rfloor$ mobile guards, each with only local views, can "self-deploy" at vertices to achieve the art gallery coverage. Their result raises the interesting question whether this gap is inherently due to the lack of global geometry. We show that this is not so, and in fact $\lfloor n/3 \rfloor$ guards in our minimal sensing model can self-deploy to guard the polygon. Unlike (Ganguli, Cortes, & Bullo 2006), our algorithm is based on triangulation, mimicking the original proof of Fisk (Fisk 1978).

The proof of (Fisk 1978) uses the fact that a triangulation can be 3-colored: three colors can be assigned to vertices

so that no diagonal has same color at both endpoints. Then, placing the guards at the *least frequent* color vertices solves the art gallery problem: there are n vertices, so the least frequent color occurs at most $\lfloor n/3 \rfloor$ times, and since each triangle must have all three colors, every triangle is visible to some guard. Thus, to solve the art gallery problem in our sensing model, we just need to show that the triangulation computed in the previous section can be 3-colored by our robots.

Lemma 3 *In our minimal sensing model, a robot can* 3*-color the triangulation of a polygon.*

Proof: See Figure 7 for illustration. The robot begins by coloring the initial vertex v_0, from which the triangulation began, as 1. It then colors all the vertices in $\mathrm{cvv}(v_0)$ alternately as 2 and 3. Thus, each 0-edge in the visibility vector of v_0 is colored $(2, 3)$. The robot now revisits the pockets P_i in the same order as in the triangulation step, and propagates the coloring. If the pocket P_i was triangulated by the robot from the vertex v_i, and the color of v_i is 2 then the diagonals incident to v_i can be colored by alternating between colors 3 and 1, and so on. It is easy to see that this coloring succeeds. ∎

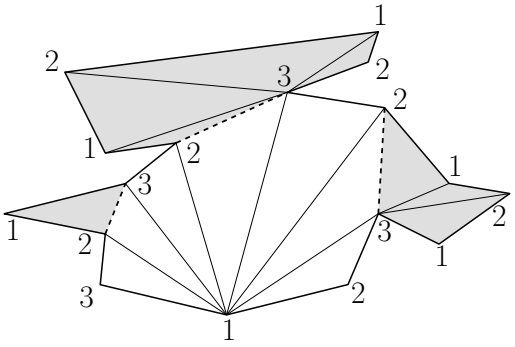

Figure 7: 3-coloring the triangulation.

Once colors have been assigned, and recorded by the robot, it can make one final tour of the polygon and determine the least frequently used color. Placing guards at those $\lfloor n/3 \rfloor$ vertices solves the art gallery problem—since the robot labels the nodes sequentially (in cyclic order) around the boundary, finding those nodes is trivial.

As before, we can implement this algorithm using a collaborative group of m robots, each with $\Theta(n/m)$ memory. In fact, once the triangulation (along with the online coloring) is in place (Theorem 5), the leader assigns every vertex of the least used color to a robot, which then can autonomously deploy.

Theorem 6 *In our minimal sensing model, a single robot with* $\Theta(n)$ *memory and a pebble can solve the art gallery problem for an* n*-vertex polygon. Alternatively, a group of* $\lfloor n/3 \rfloor$ *robots, each with* $\Theta(1)$ *memory, can solve the art gallery problem and self-deploy to guard the polygon.*

7. Conclusions

We considered a simple and minimalistic model of visibility-based robot systems, and showed that despite severe sensory limitations, these robots can accomplish fairly complex tasks, and infer global attributes of their workspace. At the same time, the impossibility of some seemingly simple topological tasks also highlights the limitations of our minimal model.

In future work, it will be interesting to explore which other global tasks are possible in this model, and which ones are not. It will also be interesting to understand the inherent power of pebbles. We showed that a single pebble suffices even for discovering all the boundary components of a multiply-connected polygon, but we do not know whether one can achieve these results without any pebble at all.

It will also be useful to study the power of additional simple primitives, such as local *angle sensing*, which circumvents the impossibility of deciding whether a vertex is convex and whether a component is the outermost boundary.

Finally, while our sensing model admits clean formulation and analysis, it obviously oversimplifies the limitations of real-world sensing: real cameras have range limitations; they are unable to distinguish vertices that are nearly collinear, etc. It will be interesting to enhance our model to include some of these non-idealities, without losing its ease of analysis.

References

Chvátal, V. 1975. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory* 18:39–41.

de Berg, M.; van Kreveld, M.; Overmars, M.; and Schwarzkopf, O. 1997. *Computational geometry: algorithms and applications*. Springer-Verlag.

Fisk, S. 1978. A short proof of Chvátal's watchman theorem. *Journal of Combinatorial Theory* 24(B):374.

Ganguli, A.; Cortes, J.; and Bullo, F. 2006. Distributed deployment of asynchronous guards in art galleries. In *Proceedings of the American Control Conference*, 1416–1421.

Guibas, L. J.; Latombe, J.-C.; LaValle, S. M.; Lin, D.; and Motwani, R. 1999. Visibility-based pursuit-evasion in a polygonal environment. *IJCGA* 9(5):471–494.

Latombe, J.-C. 1991. *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press.

Pottie, G. J., and Kaiser, W. J. 2000. Wireless integrated network sensors. *Commun. ACM* 43(5):51–58.

Sachs, S.; Rajko, S.; and LaValle, S. M. 2004. Visibility-based pursuit-evasion in an unknown planar environment. *International Journal of Robotics Research* 23(1):3–26.

Tovar, B.; Freda, L.; and LaValle, S. M. 2007. Using a robot to learn geometric information from permutations of landmarks. *Contemporary Mathematics, to appear*.

Yershova, A.; Tovar, B.; Ghrist, R.; and LaValle, S. M. 2005. Bitbots: Simple robots solving complex tasks. In *AAAI National Conference on Artificial Intelligence*.