COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 27, 167–176 (1984)

Stationing Guards in Rectilinear Art Galleries

HERBERT EDELSBRUNNER,* JOSEPH O'ROURKE,† AND EMMERICH WELZL*

*Technische Universität Graz, Institut für Informationsverabeitung, Schiesstattgasse 4a, A-8010 Graz, Austria, and †Johns Hopkins University, Department of Electrical Engineering and Computer Science, Baltimore, Maryland 21218

Received March 20, 1983; accepted May 5, 1983

A rectilinear polygon can be viewed as an art gallery room whose walls meet at right angles. An algorithm is presented that stations guards in such a room so that every interior point is visible to some guard. The algorithm partitions the polygon into L-shaped pieces, a subclass of star-shaped pieces, and locates one guard within each kernel. The algorithm runs in $O(n \log n)$ time in the worst case for a polygon of n vertices.

1. INTRODUCTION

Klee posed the following "Art Gallery" problem in 1973 [5]: how many guards are always sufficient (and sometimes necessary) to see every point in the interior of an n-wall art gallery room? The room is a simple polygon (without holes), and the guards are stationary points who can see any point connected to them by a line segment that does not go outside the polygon. Chvátal solved Klee's problem by showing that $\lfloor n/3 \rfloor$ guards are always sufficient [2]. Note that this number is often less than optimal (for example, a convex n-gon only needs one guard), but as it is necessary for at least one polygon for each n (see Fig. 1a), no smaller function of n can be sufficient for arbitrary n-gons.

This problem is of interest not only for its elegant mathematical content, but also for its practical import: for its closely related to both polygon visibility questions and polygon decomposition problems.[1] The relationship to the former is clear. The connection to decomposition can be seen by observing that each guard can see a star-shaped region (one that contains at least one point that sees the entire interior). Thus a placement of guards represents a cover of the polygon with star-shaped pieces.

Since Chvátal's proof, there have been several developments on Art Gallery problems, both mathematical and algorithmic. The mathematical developments have consisted in alternate proofs, generalizations, and specializations of the original problem. Fisk found a simpler proof of Chvátal's theorem that more easily lends itself to implementation [3, 6]. Kahn *et al.* specialized the problem to rectilinear art galleries (those consisting of only horizontal and vertical edges), and proved that $\lfloor n/4 \rfloor$ guards are always sufficient [7]; necessity follows from Fig. 1b. A new proof of this theorem was recently discovered [11]. Finally, the problem has been generalized to guards who are permitted to patrol a fixed interior line segment: then $\lfloor n/4 \rfloor$ are sufficient for simple polygons [10].

Fisk's proof for simple polygons was used by Avis and Toussaint to develop an $O(n \log n)$ algorithm for locating the $\lfloor n/3 \rfloor$ stationary guards in a simple polygon,

[1] Both areas are nicely surveyed in [13].

0734-189X/84 $3.00

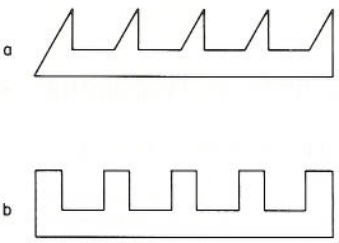

FIG. 1. Generic examples that require a maximal number of guards for coverage: (a) needs $\lfloor n/3 \rfloor$ and (b) needs $\lfloor n/4 \rfloor$ guards.

simultaneously partitioning the polygon into $\lfloor n/3 \rfloor$ star-shaped pieces [1]. This remains the fastest known algorithm for star decomposition of arbitrary simple polygons. Recently Sack presented an algorithm for partitioning a rectilinear polygon into convex quadrilaterals, and together with the proof in [7], this leads to an algorithm that locates $\lfloor n/4 \rfloor$ guards in a rectilinear gallery, simultaneously partitioning it into star-shaped pieces. His algorithm also requires $O(n \log n)$ time.

The purpose of this paper is to present a different algorithm for stationing guards in a rectilinear art gallery based on the ideas in the alternate proof of O'Rourke [11]. The algorithm, like Sack's, runs in $O(n \log n)$ time. We feel our algorithm is of independent interest, however, because Sack was primarily concerned with partitioning a rectilinear polygon into convex quadrilaterals, as was proven possible by Kahn *et al.* [7]. If only guard location (and star-shaped decomposition) is required, then the complex diversion through convex quadrilateralization can be avoided, and a more direct algorithm can be achieved.

2. SUMMARY OF PROOF OF RECTILINEAR ART GALLERY THEOREM

The algorithm to be presented in Sections 3 and 4 is based on the ideas of the proof in [11], and therefore it is necessary to repeat the skeleton of that proof here. The proof depends on three observations.

Observation 1. The total number of vertices n in a rectilinear polygon has a fixed relationship to the number of reflex vertices r: $n = 2r + 4$. (A vertex is *reflex* if the internal angle is $> 180°$; each reflex vertex in a rectilinear polygon has an internal angle of 270°.) Thus the upper bound of $\lfloor n/4 \rfloor$ proved in [7] is identical to the bound of $\lfloor r/2 \rfloor + 1$.

Observation 2. A rectilinear polygon of r reflex vertices can be partitioned into two pieces by a "cut" at one of its reflex vertices v that extends either the horizontal or vertical edge incident to v until it first intersects the boundary (see Fig. 2). Such a cut "resolves" the reflexivity at v; thus if the number of reflex vertices in the two resulting pieces are r_1 and r_2, then $r = r_1 + r_2 + 1$. If either r_1 or r_2 (or both) is odd, then recursive application of the formula $\lfloor r/2 \rfloor + 1$ achieves the correct total number of guards:

$$\lfloor r_1/2 \rfloor + 1 + \lfloor r_2/2 \rfloor + 1 = \lfloor r/2 \rfloor + 1.$$

Observation 3. Call a cut that leaves at least one of the two pieces with an odd number of reflex vertices an *odd cut*. Then every rectilinear polygon in general

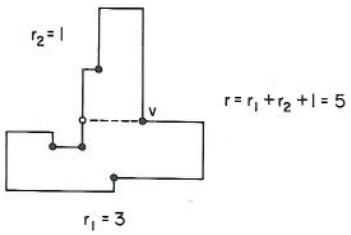

FIG. 2. A horizontal odd cut resolving the reflex vertex at v and partitions the polygon into two pieces.

position has an odd cut. Here "general position" means that no two reflex vertices can see one another along an internal horizontal or vertical line. This restriction will be discussed in Section 4.7.

The proof of this last observation is nontrivial, comprising the bulk of [11]. Here we will only need one detail from this proof: if a polygon has no *horizontal* odd cut (in which case it therefore must have a vertical odd cut), then the polygon is monotonic with respect to the horizontal axis. This structural restriction will be more fully described in Section 4.5.

3. OVERVIEW OF THE ALGORITHM

At a coarse level, the guard placement algorithm is very simple. First the polygon is preprocessed to detect which horizontal cuts are odd cuts. Then it is partitioned at every such horizontal odd cut. Finally, guards are placed in each of the resulting pieces. Observation 2 guarantees that this will achieve coverage with $\lfloor r/2 \rfloor + 1$ guards.

The simplicity of this procedure results from care applied at two critical junctures. First, it is easier algorithmically to make an odd cut in a polygon with an odd number r of reflex vertices than in one with an even number. The reason, which will be detailed in Section 4.4 below, is that no updating of the preprocessed computations are necessary for each cut when the number of reflex vertices is odd. We therefore take the counterintuitive step of complicating the polygon by introducing a new reflex vertex if r is even; the bound of $\lfloor r/2 \rfloor + 1$ is clearly unaffected.

The second critical juncture arises when all horizontal odd cuts have been made, and only vertical odd cuts remain. As previously mentioned, the polygon must then have a restricted structure, and guard placement is nearly trivial. This is fortunate, as otherwise the preprocessing step might have to be repeated with each oscillation between horizontal and vertical cuts.

These claims are justified in the following section.

4. THE ALGORITHM

The algorithm for locating $\lfloor r/2 \rfloor + 1$ guards in a rectilinear polygon of n vertices, r of which are reflex, consists of six distinct steps:

(1) If r is even, then add an additional reflex vertex.

(2) Perform a plane sweep to find all horizontal cuts.

(3) Traverse the boundary once, labeling the parity of the cuts.

(4) Partition the polygon at each horizontal odd cut.

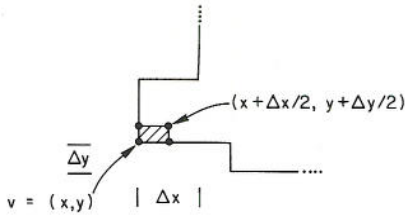

FIG. 3. A reflex vertex is introduced by removing a "chip" from the polygon at a convex vertex v. The chip is chosen to have half the length and width of the distances to the nearest vertices to v.

(5) For each resulting piece, place a guard at every other reflex vertex.

(6) Remove the extra reflex vertex, if introduced in step 1.

Each of these steps will now be described in detail and justified.

4.1. *Add Reflex Vertex: O(n)*

If r is even, then $\lfloor (r+1)/2 \rfloor + 1 = \lfloor r/2 \rfloor + 1$, so the addition of a reflex vertex is justified. The reason for doing so was alluded to above and will be expanded on below in Section 4.4. The extra reflex vertex can be added in linear time as follows. Choose an arbitrary convex vertex $v = (x, y)$. Find the smallest nonzero horizontal and vertical separations Δx and Δy from x and y to other vertices by examining the coordinates of each of the $O(n)$ vertices. Delete vertex v and replace it with three others as illustrated in Fig. 3. Since no edges of the polygon can cross the rectangle with corners (x, y) and $(x + \Delta x/2, y + \Delta y/2)$, clearly this "dent" maintains the simplicity of the polygon. That it does not interfere with visibility will be shown in Section 4.6

4.2. *Plane Sweep for Horizontal Cuts: O(n log n)*

This is the least novel aspect of the algorithm, and consequently will not be described in detail. Each reflex vertex determines a unique horizontal cut. The goal of this step is to find each horizontal cut and to insert a new "artificial" vertex into the circular list of vertices representing the polygon at the end of each cut, doubly linking each reflex vertex with its associated artificial vertex. This data structure then serves as input to step 3 of the algorithm.

The location of the artificial vertices are found by a sweep of a horizontal line from top to bottom over the polygon. This is a standard plane sweep, and is nearly identical to the monotone decomposition algorithm of Lee and Preparata [9]; see also [12]. The vertical edges of the polygon are sorted by their maximum y coordinate. At each position of the sweep line H, a data structure S holds all those vertical edges pierced by H, organized left to right. When H moves down and encounters a reflex vertex v, the vertical edge hit by v's horizontal cut is available in S as adjacent to the vertical edge whose top or bottom is v (see Fig. 4). After computing the coordinates of the corresponding artificial vertex, a vertical edge is either inserted or deleted from the data structure S, depending on whether v is the top or the bottom of a vertical edge.

The data structure can be chosen to support $O(\log n)$ insertion and deletion time; for example, either an AVL tree or a 2–3 tree [8] would be appropriate. Since there

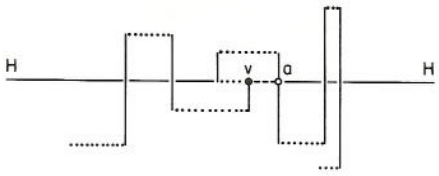

FIG. 4. The horizontal sweep line H pierces the vertical edges shown as solid lines; these are stored in a data structure S. When H hits a reflex vertex v, the artificial vertex a on its cut lies on a vertical edge adjacent to v's edge in S.

are $O(n)$ insertions and deletions, the entire sweep can be completed in $O(n \log n)$ time. In terms of worst-case complexity, this is the only supralinear step of the algorithm.

4.3. *Boundary Traversal to Compute Cut Parity: O(n)*

The next step is to determine which of the horizontal cuts are odd cuts, that is, have an odd number of reflex vertices to one side or another. As the total number of reflex vertices r is known, finding the number to one side of a cut determines the number to the other side. The number to one side of each cut can be found in a single boundary traversal of the polygon as follows.

Distinguish three types of vertices: convex, reflex, and artificial. Start at an arbitrary vertex, initialize a counter to zero, and proceed counterclockwise around the polygon. If the next vertex is convex, do nothing; if the next vertex is artificial, label it with the counter value; if the next vertex is reflex, increment the counter value and label the vertex with this new value (Fig. 5). As soon as both end points of a cut are labeled, the number of reflex vertices k to one side is determined by the difference between the two labels. When the artificial vertex of a cut is encountered second, then the exact difference of the labels is k; when the reflex vertex is second, then k is the difference less 1 (Fig. 6)

It is actually not necessary to compute the *number* of reflex vertices to each side, as only the parity is needed. Thus each reflex and artificial vertex can be labeled as even or odd during the traversal, and the parity of the cuts determined by straightforward modification of the rules above.

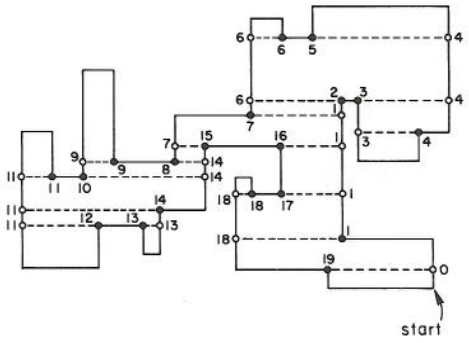

FIG. 5. Labels on reflex and artificial vertices are generated in a counterclockwise scan of the boundary, incrementing the label counter at each reflex vertex.

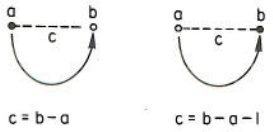

FIG. 6. The labels a and b assigned to the two ends of a cut determine the number of reflex vertices c to one side by their difference or differences less 1.

4.4. *Cut at Each Horizontal Odd Cut: O(n)*

The goal of the fourth step is to cut the polygon at each horizontal odd cut; the remaining pieces will then be easy to cover with guards. The only potential difficulty with this step is that, after a particular cut is made, the parities computed in the previous step may no longer be correct for the cuts in the two pieces. For example, Fig. 7 shows that a particular odd cut can either leave the parity of other cuts unchanged, or it could flip their parity, depending on whether the odd number of reflex vertices is inside the portion including the cut or not. However, note that the situations in Fig. 7 can only arise when the total number of reflex vertices r is even. When r is odd, then there is only one type of odd cut: a cut that has an odd number

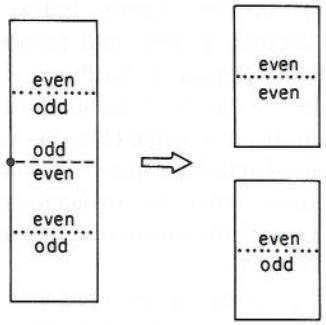

FIG. 7. Partitioning at an odd cut can either flip or leave the parity of other cuts unchanged when the total number of reflex vertices is even.

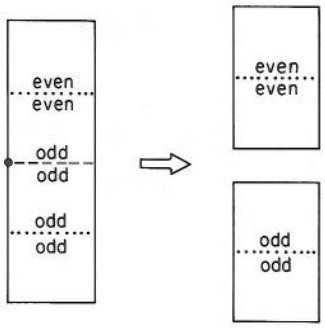

FIG. 8. If the total number of reflex vertices is odd, then partitioning at an odd cut does not affect the parity of other cuts.

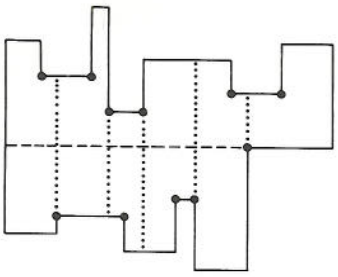

FIG. 9. Polygons that have no horizontal odd cuts have a horizontal cut (shown dashed) that partitions them into two vertical histograms. Such polygons cut vertically into L-shaped pieces at every other reflex vertex.

of reflex vertices to *each* side. Partitioning the polygon at such an odd cut leaves the parity of the cuts in each half unchanged (Fig. 8).

Since step 1 of the algorithm guarantees that r is odd, all the horizontal odd cuts may be made without any updating to the parities computed in step 3. At the conclusion of this partitioning, we are left with several rectilinear polygons, all of which have only even horizontal odd cuts remaining.

4.5. *Guard Placement: O(n)*

Consider now a rectilinear polygon with no horizontal odd cuts. If it has no reflex vertices, then it is a rectangle and can be covered with a single guard placed anywhere within the interior. This satisfies the bound $\lfloor r/2 \rfloor + 1$ of the theorem.

If there is at least one reflex vertex, then it is established in [11] that a vertical odd cut exists. However, finding such a cut would require repetition of the previous four steps of the algorithm, and ultimately may lead to switching back to horizontal cuts again. Such oscillation and the computation it entails could be very expensive. Fortunately, polygons without horizontal odd cuts have a special structure that makes guard placement easy.

Although it is by no means obvious, it is proven in [11] that a polygon with only even horizontal cuts must appear as in Fig. 9: it is formed by two "histograms" joined at their bases.[2] More precisely, define a *vertical histogram* as a rectilinear polygon that has one horizontal edge (the base) equal in length to the sum of lengths of all the other horizontal edges. Then a polygon that has no horizontal odd cuts must have a reflex vertex whose horizontal cut partitions the polygon into two vertical histograms (as in Fig. 9). Such a polygon clearly must have an odd number of reflex vertices.

Guards placed at every other reflex vertex from left to right will necessarily cover such a polygon. To see this, let the reflex vertices be $v_1, v_2, \ldots, v_k$ in sorted order from lowest x coordinate to highest; here k is odd. Make a vertical cut at $v_2, v_4, \ldots, v_{2\lfloor k/2 \rfloor}$. The polygon has now been partitioned into $\lfloor k/2 \rfloor + 1$ L-shaped pieces, each containing one reflex vertex (Fig. 9). Place guards on $v_1, v_3, \ldots, v_{2\lfloor k/2 \rfloor + 1}$. This clearly covers the polygon with $\lfloor k/2 \rfloor + 1$ guards, achieving the bound of the theorem.

[2] This structural restriction is stated in a different but equivalent terms in [11].

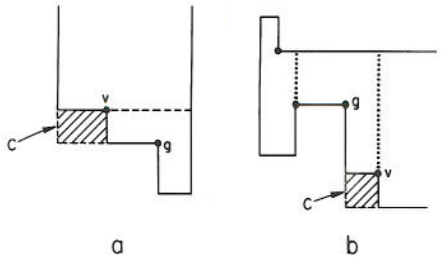

FIG. 10. The chip region C is visible to some guard g, regardless of whether a horizontal (a) or vertical (b) cut emanates from v.

The sorting of the reflex vertices can be accomplished in linear time by merging the two histogram chains, which are necessarily already sorted. Once the order is determined, no vertical cuts need actually be made: the guards are simply assigned to every other reflex vertex. Note that after step 5 is completed, the polygon has been implicitly partitioned into L-shaped pieces; one of these may be modified to a rectangle in the final step below.

Finally we note that the same simple guard location procedure would work if the polygon were *monotone* with respect to the x axis. In such a polygon the boundary can be partitioned into two chains such that the x coordinates of the vertices in each chain increase or decrease monotonically [9]. This is a wider class of shapes than the double histograms that result from step 5.

4.6 Replacement of Chip: O(n)

If an extra reflex vertex was introduced by removing a "chip" C at a convex corner of the polygon in step 1 of the algorithm (Fig. 10), then returning the polygon to its original form will not require any more guards, nor even a movement of the existing guard locations.

Consider two cases. If the introduced reflex vertex v is assigned a guard during step 5, then leave the guard at that position and retract the chip back to the original convex corner. Clearly everything is covered. If v is not assigned a guard, then a cut was made at that vertex in the algorithm, either a horizontal cut in step 3 or a vertical cut (implicitly) in step 4. In either case, one of the two edges incident to v is a complete edge of an L-shaped region that has a guard g in it (Fig. 10). This follows since the edges of the chip were chosen to be too small to be hit by any cuts. Clearly the guard g sees into C.

4.7. *Degenerate Cases*

We have assumed that no two reflex vertices of the polygon can see one another along a vertical or horizontal line. Observation 3 depends on this assumption: without it a polygon may have no odd cuts, either horizontal or vertical [11]. Fortunately this degeneracy is in our favor, so to speak, and is easy to handle.

If two reflex vertices see one another along a horizontal, this can be detected during the plane sweep, step 2 of the algorithm. Before commencing step 3, we can cut the polygon into pieces at each such horizontal. Each cut resolves two reflex vertices, so that $r = r_1 + r_2 + 2$, where r_1 and r_2 are the number of reflex vertices in

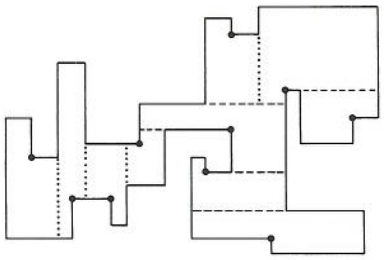

FIG. 11. Guards positioned in the polygon of Fig. 5. The dashed lines are the horizontal cuts made in step 4 of the algorithm, and the dotted lines are the vertical cuts implicit in step 5. Here $r = 19$, and $\lfloor r/2 \rfloor + 1 = 10$ guards are used.

the two resulting pieces. Regardless of the parity of r_1 and r_2,

$$\lfloor r_1/2 \rfloor + 1 + \lfloor r_2/2 \rfloor + 1 \leq \lfloor (r_1 + r_2)/2 \rfloor + 2 = \lfloor r/2 \rfloor + 1.$$

Recursive application therefore achieves the desired bound, and each piece can be processed by the algorithm separately.

If two reflex vertices see one another along a vertical, one of the L-shaped regions formed by the implicit vertical cuts in step 5 may degenerate to a rectangle, but this in no way affects the execution of the analysis of the algorithm.

5. DISCUSSION

The guard locations chosen by the algorithm for the example used in Fig. 5 and the partition they induce, are shown in Fig. 11. Our algorithm can be viewed as solving several problems: guard location, star-shaped decomposition, or partitioning into L-shaped pieces. The algorithm runs in $O(n)$ time if the polygon is monotone (a condition that can be checked in linear time [12]), but requires $O(n \log n)$ time for arbitrary rectilinear polygons. The entire algorithm would run in linear time if the horizontal cuts (step 2) could be computed in linear time. This problem of finding all horizontal cuts shares some similarity with the problem of triangulating a simple polygon in linear time, a long standing open problem in computational geometry [4]. It may be that the "rectilinear cutting problem" is more approachable, either for a linear-time solution or an $O(n \log n)$ lower bound proof.

REFERENCES

1. D. Avis and G. Toussaint an efficient algorithm for decomposing a polygon into star-shaped polygons, *Pattern Recognition* **13**, 1981, 395–398.
2. V. Chvátal, A combinatorial theorem in plane geometry, *J. Combinatorial Theory B*, **18**, 1975, 39–41.
3. S. Fisk, A short proof of Chvátal's watchman theorem, *J. Combinatorial Theory B*, **24**, 1978, 374.
4. M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan, Triangulating a simple polygon, *Inform. Processing Lett.* **7**, 1978, 175–179.
5. R. Honsberger, *Mathematical Gems II*, pp. 104–110, Mathematical Association of America, 1976.
6. R. Honsberger, Games, graphs, and galleries, pp. 274–284, in *The Mathematical Gardner*, ed. (David A. Klarner, Ed.), Prindle, Weber & Schmidt, Boston, 1981.
7. J. Kahn, M. Klawe and D. Kleitman, Traditional galleries require fewer watchman, *Siam J. Alg. Disc. Meth.* **4**, 1983, 194–206.
8. D. E., Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, pp. 451–469, Addison–Wesley, 1973.

9. D. T. Lee, and F. P. Preparata, Location of a point in a planar subdivision and its applications, *SIAM J. Comput.* **6**, 1977, 594–606.
10. J. O'Rourke, Galleries need fewer mobile guards: A variation on Chvátal's theorem, *Geometriae Dedicata* **14**, 1983, 273–283.
11. J. O'Rourke, An alternate proof of the rectilinear art gallery theorem, *J. Geometry*, in press.
12. Jörg-R. Sack, An $O(n \log n)$ Algorithm for Decomposing Simple Rectilinear Polygons into Convex Quadrilaterals, Pro. Twentieth Allerton Conference, Monticello, Illinois, pp. 64–74, Oct. 1982.
13. G. T. Toussaint, Pattern Recognition and Geometrical Complexity, Proc. of 5th International Conf. on Pattern Recognition, Miami Beach, Florida, pp. 1324–1347, Dec. 1980.