TESTING A SIMPLE POLYGON FOR MONOTONICITY *

Franco P. PREPARATA
Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801, U.S.A.

Kenneth J. SUPOWIT
Department of Computer Science, University of Illinois, Urbana, IL 61801, U.S.A.

Received 10 January 1981; revised version received 26 May 1981

Computational geometry, monotone polygon, analysis of algorithms

1. Introduction

Let P be a simple polygon in the plane having vertices $p_0, p_1, ..., p_{n-1}$ counterclockwise on its boundary. The sides of P, called *arcs*, are denoted as $e_j = (p_j, p_{j+1})$ and are directed from p_j to p_{j+1} (indices are taken modulo n throughout). A *chain* $C_{ij} = (e_i, e_{i+1}, ..., e_{j-1})$ is a sequence of arcs on the boundary of P. C_{ij} is *monotone with respect to* a (straight) line ℓ if the projections of the vertices $p_i, p_{i+1}, ..., p_j$ on ℓ are ordered as the vertices in C_{ij}. P is *monotone* if there exists a line ℓ such that the boundary of P can be partitioned into two chains C_{ij} and C_{ji} that are monotone with respect to ℓ (if a direction is chosen on ℓ then one chain is monotone non-decreasing, the other is monotone non-increasing).

Note that the class of monotone polygons properly contain the class of convex polygons, and are properly contained in the class of simple polygons. It appears that certain computational problems involving polygons are easier for monotone than for arbitrary simple polygons. For example, the fastest algorithm known to triangulate an arbitrary simple polygon requires $\Theta(n \log n)$ time [1]; by 'time' in this paper, we mean worst-case time. However, given a line ℓ and a polygon P monotone with respect to ℓ, P can be triangulated in $\Theta(n)$ time [1].

We consider the following problem: given a simple polygon P, decide whether P is monotone and, if so, exhibit a line ℓ with respect to which P is monotone. We present a $\Theta(n)$ time solution to this problem; hence, by the above remarks, there is a $\Theta(n)$ time algorithm that, given a simple polygon, triangulates it in $\Theta(n)$ time if it is monotone.

2. The algorithm

Given the polygon P, as defined in the preceding section, let θ_i be the counterclockwise polar angle at arc e_i ($i = 0, ..., n-1$) with respect to a chosen direction (for example, the direction of e_0). Define α_i as the counterclockwise *wedge* from θ_{i-1} to θ_i if the external angle at vertex p_i is $\geqslant 180°$; as the clockwise wedge from θ_{i-1} to θ_i, otherwise. Note that, by the simplicity of P, the angle of wedge α_i ($i = 0, ..., n-1$) has size $< 180°$.

Given a chain $C = (e_j, e_{j+1}, ..., e_{j+k})$, define $\alpha(C) \triangleq \bigcup_{i=j+1}^{j+k} \alpha_i$, i.e., $\alpha(C)$ is the union of the wedges $\alpha_j, \alpha_{j+1}, ..., \alpha_{j+k-1}$. Obviously, $\alpha(C)$ is a wedge. We now prove:

Lemma 1. $C = (e_1, ..., e_k)$ is monotone with respect to ℓ if and only if the normal to ℓ has a polar angle $\theta \notin \alpha(C)$.

* This work was supported in part by National Science Foundation Grant MCS-78-13642, by the Joint Services Electronics Program Contract N00014-79-C-0424, and by Control Data Corporation Contract CDCI-04-AB.

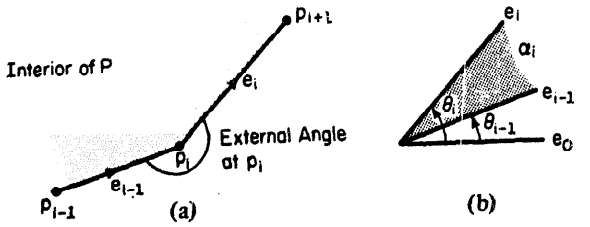

Fig. 1. Illustration of the correspondence between wedge α_i and the external angle at p_i.

Proof. Given that C is monotone with respect to ℓ, suppose that $\theta \in \alpha(C)$ (Fig. 2b). This implies that there is at least one wedge α_i such that $\theta \in \alpha_i$, for some $i \in \{1, 2, ..., k-1\}$. If we now consider (Fig. 2a) the projections of vertices p_i, p_{i+1}, and p_{i+2} on ℓ, we have that the projections of p_i and p_{i+2} are on the same side of that of p_{i+1}, i.e., C is not monotone with respect to ℓ, a contradiction. Thus $\theta \notin \alpha(C)$.

Conversely, suppose that C is not monotone with respect to ℓ. Then there is a vertex p_i of C for which the preceding arguments can be reversed.

Consider now a monotone polygon P. Monotonicity means that there are two vertices p_i and p_j of P and a line ℓ, such that chains C_{ij} and C_{ji} are monotone with respect to ℓ (Fig. 3a). In the polar diagram (Fig. 3b), we construct the wedges $\alpha(C_{ij})$ and $\alpha(C_{ji})$. These two wedges are possibly separated by two wedges γ_j and γ_i (see Fig. 3b). Note that $\theta_{i-1} \in \alpha(C_{ji})$ and $\theta_i \in \alpha(C_{ij})$; also $\gamma_i \subseteq \alpha_i$, whence size $(\gamma_i) \leqslant$ size $(\alpha_i) < 180°$. Similarly, size $(\gamma_j) < 180°$. Moreover, by Lemma 1, the line ℓ' passing through the origin of the polar diagram and perpendicular to ℓ intersects neither

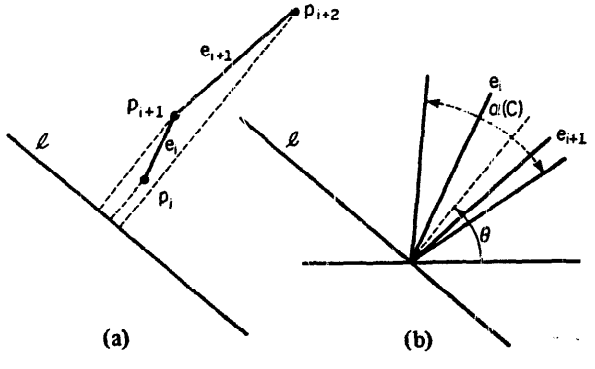

Fig. 2. Illustration for the proof of Lemma 1.

$\alpha(C_{ij})$ nor $\alpha(C_{ji})$; thus $\alpha(C_{ij})$ and $\alpha(C_{ji})$ lie on opposite sides of ℓ'.

The polar rays corresponding to the n arcs of P partition the polar range $[0, 2\pi)$ into n consecutive wedges (some of which could be of size 0). Let α be one of these wedges; the *multiplicity* $\mu(\alpha)$ of α is defined as $\mu(\alpha) \triangleq |\{\alpha_i: \alpha \subseteq \alpha_i\}|$, i.e., the number of wedges α_i which contain α. It follows that the previously introduced γ_i and γ_j are precisely wedges whose multiplicity is 1 and which are *antipodal* (i.e., their union wholly contains a straight line passing through the pole). It is not difficult to see that the arguments can be reversed, thus proving the following theorem:

Theorem. A simple polygon P is monotone if and only if the polar diagram of its arcs contains at least one pair of antipodal intervals of multiplicity 1.

This theorem immediately suggests an algorithm to test a simple polygon P for monotonicity: we process the boundary of P in the order $e_0, e_1, ..., e_{n-1}$. When we process an edge e_j, we insert α_j into the polar diagram by updating the multiplicities of the polar wedges so far constructed. Note that the multiplicity of a wedge cannot decrease; since we are seeking polar wedges of multiplicity 1, it is irrelevant whether a wedge has multiplicity 2 or greater. Thus we shall label each wedge with a symbol in the set {0, 1, 2}, where {0, 1} are actual multiplicities and 2 denotes a multiplicity $\geqslant 2$.

During processing we maintain a doubly-linked circular list of polar angles, each of which separates two adjacent wedges. Each of the two pointers (forward and backward) is labeled in the set {0, 1, 2}. In addition, we have a pointer to the current position θ in the polar diagram. We claim that the list satisfies the following properties:

(1) the angles are in increasing counterclockwise order;

(2) the wedge labels – possibly with the exception of one single 0 label – form an alternating string of 1's and 2's.

To prove this claim, we outline the algorithm.

Initial step. 0 is chosen conventionally as θ_0. There is a single wedge, labeled 0. We insert into the list angle θ_1 and label with 1 the wedge determined by α_1, and θ is set to θ_1.

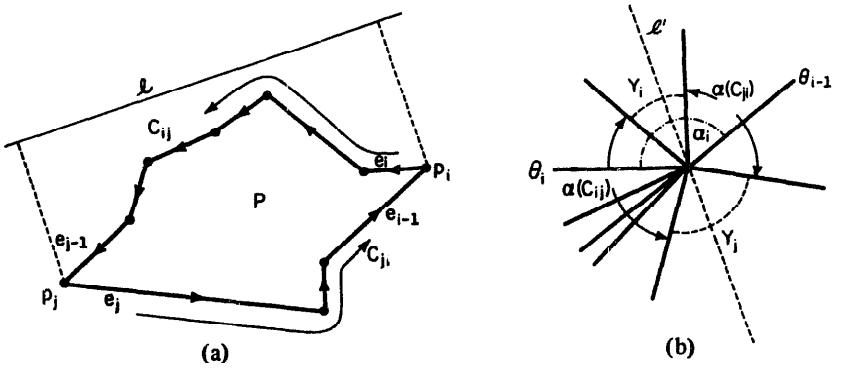

Fig. 3. Illustration of the correspondence between a simple polygon P and polar diagram of its arcs.

General step. Let θ be the current position and assume that the list satisfies properties (1) and (2). We process α_i. If θ_i is larger than θ we scan the list forward, while if θ_i is not larger than θ we scan it backwards. The scan terminates when θ_i can be inserted. In this process we increase by 1 each wedge label different from 2 and merge any two consecutive wedges receiving identical labels (merging is, of course, done by deleting the node corresponding to the angle value which separates them). With regard to the updating of θ, suppose that θ_i is to be inserted into wedge $[\beta, \beta']$: if the pointer from β to β' is labeled 0 or 1, then a new list node is created and $\theta \leftarrow \theta_i$; else no new node is created and $\theta \leftarrow \beta'$.

Clearly property (1) is satisfied after the general step, because θ_i is inserted in its appropriate order. Property (2) is also satisfied, since wedge merging guarantees the alternation of 1 and 2 labels on contiguous wedges (with labels different from 0).

From the performance viewpoint, it is convenient to charge the computational work to each individual list node. A list node is initially established in constant time. Subsequently, during list scans, a node is traversed in one direction; its pointers are for brevity referred to as *incoming* and *outgoing*. The labels of both pointers are updated ($0 \rightarrow 1$, $1 \rightarrow 2$, $2 \rightarrow 2$) and when both pointers are labeled 2 the node is deleted. Each node traversal uses constant time and each node can be traversed at most twice before its deletion. It follows that the total running time is O(n).

At the termination of the above algorithm, we have a partition of the polar range $[0, 2\pi)$ into O(n) wedges with alternating labels 1 and 2. Scanning the sequence of angles by means of two pointers b_1 and b_2 we can determine the pairs of antipodal wedges. [1] Specifically, let $\theta(b_1)$ denote the angle pointed to by b_i. We set initially $\theta(b_i) = 0$ and advance $\theta(b_2)$, until $\theta(b_2) - \theta(b_1) \geqslant 180°$; at this point $\theta(b_1)$ is advanced until $\theta(b_2) - \theta(b_1) < 180°$, when the advancement of $\theta(b_2)$ is resumed; and so on until $\theta(b_2) = 0$. This process clearly runs in time O(n) and obtains all pairs of antipodal wedges (which are known to be O(n) [3]), whose labels are concurrently compared. Since both major tasks (construction of the sectors and detection of antipodal pairs) can be completed in linear time, the entire monotonicity test runs in linear time, which is optimal.

Note that the above algorithm obtains *all* directions with respect to which P is monotone.

Conclusion

Testing an arbitrary polygon (i.e., a sequence of vertices) for convexity [3], testing a simple polygon for star-shapedness [2], and testing a simple polygon for monotonicity are all Θ(n) time problems. An interesting open problem in this area is testing an arbitrary polygon for simplicity. For this problem, the fastest algorithm known is Θ(n log n) time [4], but no superlinear lower bound is known.

[1] The following technique is a modification of an algorithm due to M.I. Shamos [3] to obtain the diameter of a convex polygon.

Acknowledgements

The authors thank G.T. Toussaint and H. El-Gindy for helpful discussions on this problem.

References

[1] M.R. Garey, D.S. Johnson, F.P. Preparata and R.E. Tarjan, Triangulating a simple polygon, Information Processing Lett. 7 (4) (1978) 175–179.

[2] D.T. Lee and F.P. Preparata, An optimal algorithm for finding the kernel of a polygon, J. ACM 26 (1979) 415–421.

[3] M.I. Shamos, Computational geometry, Ph.D. thesis, Dept. of Computer Science, Yale University (1978).

[4] M.I. Shamos and D. Hoey, Geometric intersection problems, 17th Annual Symposium on Foundations of Computer Science (1976) 208–215.