A Generalization of Staircase Visibility *

Sven Schuierer† Gregory J. E. Rawlins ‡ Derick Wood §

Abstract

In this paper we investigate the consequences of defining visibility based on special classes of curves rather than straight lines. The general framework we make use of here is known as $\mathcal{O}$-convexity, a generalization of the usual notion of convexity in the plane.

We investigate the structural properties of $\mathcal{O}$-convex curves in simple polygons. Further, we give algorithms to construct the kernel under various versions of visibility for simple polygons.

1 Introduction

The *kernel* of a polygon is the set of all points which see every point in the polygon. Finding the kernel of a polygon is one of the central visibility problems in computational geometry.

Visibility questions can also be seen as *reachability* questions. The particular problem we have in mind is that of "guarding" a polygon with one robot whose motion is restricted in some way. The particular restriction we investigate here is that of requiring that the robot's path must be *monotone* in some set of orientations.

In [6] we motivated and gave a new definition of "sight" different to the usual notion of straight line visibility. Coupled with the convexity considerations from [5] this lead us to general visibility considerations.

2 Definitions

We shall assume that we are working in the plane and that the plane is oriented in the usual way. Given a directed line l we say that its *orientation* is the counterclockwise angle made with the positive x-axis and we denote it by $\Theta(l)$. Given an undirected line we say that its *orientation* is the smaller of the two possible orientations.

A curve C is *x-convex* if the intersection of C with any horizontal line is connected. Equivalently, C is *x-convex* if it is y-monotone. This definition can be extended to *θ-convex* curves by choosing the class of intersecting lines as the lines of orientation θ instead of the lines parallel to the x-axis (the lines of orientation $0°$).

We further generalize this notion to a set of orientations as follows: Given a set of orientations $\mathcal{O}$ we say that the curve C is *$\mathcal{O}$-convex* if it is θ-convex, for all θ in $\mathcal{O}$. It is easy to see that if $\mathcal{O}^{\perp}$ denotes the set of orientations which are orthogonal to those in $\mathcal{O}$, then a curve C is $\mathcal{O}$-convex if and only if C is monotone with respect to all orientations in $\mathcal{O}^{\perp}$. Note that if $\mathcal{O}$ is the set of *all* orientations, the only $\mathcal{O}$-convex curves are lines, rays, and line segments. From now on we will call an $\mathcal{O}$-convex curve an *$\mathcal{O}$-staircase segment.*

Since we defined $\mathcal{O}$-convexity using the intersection of lines, we can assume that with θ we always have $\theta + 180°$ in $\mathcal{O}$. We denote $\theta + 180°$ by θ^{-1}.

3 Restricted Orientation Visibility

Our last observation suggests that we define the generalized notion of visibility based on the concept of $\mathcal{O}$-staircase segments.

Definition 3.1 *Let* P *be a polygon. We say two points p and q in* P $\mathcal{O}$-see *each other or are* $\mathcal{O}$-visible *from each other if there is an $\mathcal{O}$-staircase segment in* P *that connects p and q.*

*This work was supported by the Deutsche Forschungs Gemeinschaft under Grant No. Ot 64/5–4 and by Natural Sciences and Engineering Research Council Grant No. A-5692.

†Institut für Informatik, Universität Freiburg, Rheinstr. 10–12, D-7800 Freiburg, Fed. Rep. of Germany.

‡Department of Computer Science, Indiana University, 101 Lindley Hall, Bloomington, Indiana 47405, USA. email: rawlins@iuvax.cs.indiana.edu

§Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L3G1, Canada. email: dwood%watdaisy@waterloo.csnet

Apart from the usual definition of straight line visibility which is captured by the special case that $\mathcal{O} = [0°, 360°)$ the case that $\mathcal{O} = \{0°, 90°\}$ has also been studied in the literature [2,4]. This is also known as *staircase-visibility*.

In the following we are mainly interested in algorithms to compute the $\mathcal{O}$-kernel which is defined below. We will restrict ourselves to simple polygons and finite $\mathcal{O}$.

Definition 3.2 *The* $\mathcal{O}$-kernel *of a polygon* P *is the set of points in* P *which* $\mathcal{O}$*-see all other points in* P.

3.1 $\mathcal{O}$-Visibility and the $\mathcal{O}$-kernel

Before we can give an algorithm to compute the $\mathcal{O}$-kernel of a simple polygon, we need the following two powerful results about $\mathcal{O}$-visibility.

Lemma 3.1 *Let* $\mathcal{O}_i$, $i \in I \subseteq \mathbb{N}$, *be some family of orientations and* $\mathcal{O}$ *their union, i.e.* $\mathcal{O} = \bigcup \mathcal{O}_i$. *If* $\mathbf{P}$ *is a simple polygon and* p *and* q *are two points in* $\mathbf{P}$, *then we have that*

$$p \text{ } \mathcal{O}\text{-sees } q \text{ if and only if } p \text{ } \mathcal{O}_i\text{-sees } q, \text{ for all } i \in I.$$

As an immediate consequence we obtain the following relationship between the $\mathcal{O}$-kernel and the $\mathcal{O}_i$-kernel of a polygon.

Corollary 3.2 *Let* $\mathcal{O}_i$, $i \in I \subseteq \mathbb{N}$, *be some family of orientations and* $\mathcal{O}$ *their union. If* $\mathbf{P}$ *is a simple polygon, then*

$$\mathcal{O}\text{-}kernel(\mathbf{P}) = \bigcap_{i \in I} (\mathcal{O}_i\text{-}kernel(\mathbf{P})).$$

This immediately yields an algorithm to compute the $\mathcal{O}$-kernel of a simple polygon, for finite $\mathcal{O}$, once we are able to compute the $\mathcal{O}$-kernel for one direction since Corollary 3.2 implies that

$$\mathcal{O}\text{-}kernel(\mathbf{P}) = \bigcap_{\theta \in \mathcal{O}} (\{\theta\}\text{-}kernel(\mathbf{P})).$$

3.2 The $\{0°\}$-Kernel of a Polygon

In this subsection we consider the special case that $|\mathcal{O}| = 1$. W.l.o.g. we can assume that $\mathcal{O} = \{0°\}$. Given a polygon $\mathbf{P}$, we wish to determine $\{0°\}$-*kernel*$(\mathbf{P})$. That is, the set of all points which see every point in $\mathbf{P}$ via $\{0°\}$-stairsegments which lie in $\mathbf{P}$. Recall that a $\{0°\}$-stairsegment is a curve which is monotone with respect to the y-axis. We develop an algorithm to find $\{0°\}$-*kernel*$(\mathbf{P})$ in linear time and space. We use the usual RAM model of computation in which any arithmetic operation costs constant time.

Consider a reflex vertex v_i in $\mathbf{P}$ where v_{i-1} and v_{i+1} are either both above or both below v_i. These vertices create two types of "extrema"—*reflex maxima* and *reflex minima*. Note that a horizontal edge with two reflex vertices may also form a reflex maximum or minimum.

Lemma 3.3 *Given a simple polygon* $\mathbf{P}$, $\{0°\}$-*kernel*$(\mathbf{P})$ *lies between the lowest reflex minimum and the highest reflex maximum.*

Proof: We only prove that no point above the lowest reflex minimum or below the highest reflex maximum can belong to $\{0°\}$-*kernel*$(\mathbf{P})$. To this end consider the horizontal line l through a reflex maximum v_i. If there is a point p in $\{0°\}$-*kernel*$(\mathbf{P})$ that lies below v_i, then either v_{i-1} or v_{i+1} cannot be $\{0°\}$-seen from p since any curve crossing l twice cannot be $\{0°\}$-convex. □

The above lemma immediately yields an algorithm to compute $\{0°\}$-*kernel*$(\mathbf{P})$. We just have to find the lowest reflex minimum v_l and highest reflex maximum v_h and then output the left and right part of the boundary of $\mathbf{P}$ with a y-range in the interval $[v_h, v_l]$. All of this can be done in linear time. So we have the following result.

Theorem 3.4 *Let* $\mathbf{P}$ *be a simple polygon, then* $\{0°\}$-*kernel*$(\mathbf{P})$ *can be computed in time linear in the number of vertices.*

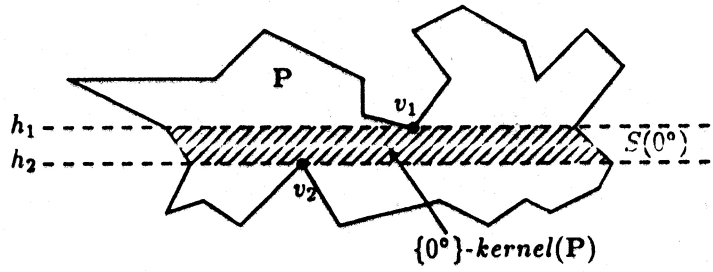

Figure 1: $\{0°\}$-*kernel*(P) is the intersection of $S(0°)$ and P.

3.3 A Fast Algorithm for the $\mathcal{O}$-Kernel of a Polygon

Above we presented an algorithm which computes the $\{\theta\}$-kernel of a polygon P, for each $\theta \in \mathcal{O}$, and then intersects the kernels obtained to yield $\mathcal{O}$-*kernel*(P). Since we have to apply the $\{0°\}$-kernel subroutine $|\mathcal{O}|$-times, the algorithm needs at least time $\Omega(n \cdot |\mathcal{O}|)$ to compute $\mathcal{O}$-*kernel*(P) even if we do not take the time for intersecting the $\{\theta\}$-kernels of P into account.

In the following we will give an algorithm to compute $\mathcal{O}$-*kernel*(P) that only takes $O(n \log |\mathcal{O}|)$ time if $\mathcal{O}$ is given as a sorted set. In order to develop an algorithm that obtains the above time bound, we have to view the $\{0°\}$-kernel a little differently. Let v_1 be the lowest reflex minimum and v_2 the highest reflex maximum of P. If we denote the horizontal line through v_i by h_i, for $i = 1, 2$, then Lemma 3.3 implies that $\{0°\}$-*kernel*(P) is the intersection of the slab $S(0°)$ between h_1 and h_2 with P (see Figure 1). Hence, if we are given the slabs $S(\theta)$ that lie between the θ-oriented lines through the lowest θ-reflex minimum and the highest reflex θ-maximum, for all $\theta \in \mathcal{O}$, then we have that

$$\mathcal{O}\text{-}kernel(\mathbf{P}) = \bigcap_{\theta \in \mathcal{O}} (S(\theta) \cap \mathbf{P}) = \left(\bigcap_{\theta \in \mathcal{O}} S(\theta) \right) \cap \mathbf{P}$$

In the following we denote $\bigcap_{\theta \in \mathcal{O}} S(\theta)$ by $\mathbf{C(P)}$. Since $S(\theta)$ is the intersection of two θ-oriented halfplanes, $\mathbf{C(P)}$ is an $\mathcal{O}$-oriented convex set. This, in particular, implies that it consists of at most $|\mathcal{O}|$ edges. Hence, $\mathbf{C(P)} \cap \mathbf{P}$ can be computed in time $O(n \log |\mathcal{O}|)$ by testing, for each edge e of P, if e intersects an edge of $\mathbf{C(P)}$. This yields the intersection points of $\mathbf{C(P)}$ with P and enables us to compute the vertices of $\mathbf{C(P)} \cap \mathbf{P}$ in additional linear time.

So we only have to show how to compute $\mathbf{C(P)}$ in time $O(n \log |\mathcal{O}|)$. Given a point p in the plane we denote the (closed) halfplane above the θ-oriented line l through p by $h^+(p, \theta)$. For a given orientation θ, we denote the highest reflex θ-maximum of P by v_θ.

The first observation we make is that $S(\theta)$ is the is the intersection of $h^+(v_\theta, \theta)$ with $h^+(v_{\theta^{-1}}, \theta^{-1})$ since the lowest reflex θ-minimum is, of course, the highest reflex θ^{-1}-maximum. Since we always have that θ^{-1} is in $\mathcal{O}$ if $\theta \in \mathcal{O}$, we only have to find the intersection of the θ-oriented halfplanes above the reflex θ-maxima, taken over all $\theta \in \mathcal{O}$. Therefore, if v_θ is the highest reflex θ-maximum of P, for each $\theta \in \mathcal{O}$, then we want to compute $\bigcap_{\theta \in \mathcal{O}} h^+(v_\theta, \theta)$.

The idea of the algorithm is to scan the boundary of P counterclockwise and, thus, successively process the vertices of P. During the scan we keep a list $\mathcal{L}$ of the highest reflex θ-maxima we have encountered so far, for each $\theta \in \mathcal{O}$. So when we process the next reflex vertex v, we check for which orientations $\theta \in \mathcal{O}$ v is a reflex θ-maximum. We replace those θ-maxima v_θ for which v is higher than v_θ in the θ-coordinate system by v. The problem is that there may be $\Omega(|\mathcal{O}|)$ of orientations θ which have v_θ replaced by v. Hence, the algorithm may still take time $\Omega(n|\mathcal{O}|)$ if there are $\Omega(n)$ reflex vertices, each being a θ-maximum for $\Omega(|\mathcal{O}|)$ orientations θ.

The important observation to reduce the running time of the algorithm is that we are only interested in the intersection of the halfplanes $h^+(v_\theta, \theta)$, with $\theta \in \mathcal{O}$. If we have that the so far encountered highest reflex θ-maximum v_θ is lower than v, for a subset $\mathcal{O}'$ of orientations θ in $\mathcal{O}$, the following lemma shows that there exist two orientations θ_1 and θ_2 such that $h^+(v, \theta_1) \cap h^+(v, \theta_2)$ already equals $\bigcap_{\theta \in \mathcal{O}'} h^+(v, \theta)$. If we can find θ_1 and θ_2 in logarithmic time, we only have to replace v_{θ_1} and v_{θ_2} in $\mathcal{L}$ since only these may influence the final intersection. If $\mathcal{L}$ is stored as a dictionary, then replacing v_{θ_1} and v_{θ_2} can be achieved in time $O(\log |\mathcal{O}|)$. Hence, it only has to be shown that θ_1 and θ_2 exist and can be found in time $O(\log |\mathcal{O}|)$.

Lemma 3.5 *If* P *is a polygon,* $\mathcal{O}$ *a finite set of orientations, and* v *a reflex vertex of* P*, then there are two*

orientations θ_1 and θ_2 in $\mathcal{O}$ such that

$$h^+(v,\theta_1) \cap h^+(v,\theta_2) = \bigcap_{\theta} h^+(v,\theta)$$

if the intersection on the right side is taken over all $\theta \in \mathcal{O}$ for which v is a reflex θ-maximum. Furthermore, v_1 and θ_2 can be found in time $O(\log |\mathcal{O}|)$.

So the scanning algorithm yields a number of vertices v_θ such that

$$\bigcap_{\theta \in \mathcal{O}} h^+(v_\theta,\theta) \cap \mathbf{P}$$

yields the $\mathcal{O}$-kernel of $\mathbf{P}$. Above we have shown that we can compute the vertices v_θ in time $O(n \log |\mathcal{O}|)$. Observe that there are at most $O(|\mathcal{O}|)$ such vertices, at most one for each $\theta \in \mathcal{O}$. Furthermore, note that the above computation yields the vertices v_θ sorted according to θ. It is easy to show that the intersection of halfplanes that are sorted according to slope can be computed in time linear in the number of halfspaces. Hence, the intersection $\bigcap_{\theta \in \mathcal{O}} h^+(v_\theta,\theta)$ can be computed in time $O(|\mathcal{O}|)$. This proves the following theorem.

Theorem 3.6 *The $\mathcal{O}$-kernel of a simple polygon with n vertices can be computed in time $O(n \log |\mathcal{O}| + |\mathcal{O}|)$, for finite $\mathcal{O}$, given $O(|\mathcal{O}| \log |\mathcal{O}|)$ preprocessing time to sort $\mathcal{O}$.*

There are two questions in connection with the above result.

(i) Is it possible to compute the $\mathcal{O}$-kernel of polygon within the same time bounds if $\mathcal{O}$ consists of a finite number of intervals, i.e. the algorithm should have a running time of $O(n \log r)$ if $\mathcal{O}$ consists of r intervals?

(ii) Is there an algorithm that computes the $\mathcal{O}$-kernel faster or can a lower bound be proven?

As to the first question we conjecture that this is possible using an approach that is similar to the kernel algorithm of Lee and Preparata [3], i.e. processing the vertices in a scan as in our algorithm and keeping track of a suitable representation of the $\mathcal{O}$-kernel.

As for the second question the situation is less clear. On the one hand, there is an easy to show lower bound on finding all vertices which are a θ-maximum, for some θ in $\mathcal{O}$, of $\Omega(n \log |\mathcal{O}|)$, on the other hand there is a kernel algorithm by Cole and Goodrich [1] which may be adapted to yield an $O(n + |\mathcal{O}|)$ time algorithm for the computation of $\mathcal{O}$-*kernel*$(\mathbf{P})$.

References

[1] Richard Cole and Michael T. Goodrich. Optimal parallel algorithms for polygon and point-set problems. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, pages 201–210, ACM, ACM Press, Champaign, Illinois, June 1988.

[2] Joseph Culberson and Robert Reckhow. *A Unified Approach to Orthogonal Polygon Covering Problems via Dent Diagrams.* Technical Report TR 89-6, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, February 1989.

[3] D. T. Lee and F. P. Preparata. An optimal algorithm for finding the kernel of a polygon. *Journal of the ACM*, 26(3):415–421, July 1979.

[4] Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: the perfect graph approach. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, pages 211–223, ACM, Urbana, Illinois, June 1988.

[5] Gregory J. E. Rawlins and Derick Wood. Computational geometry with restricted orientations. In *Proceedings of the 13th IFIP Conference on System Modelling and Optimization*, Springer Verlag, 1988. Lecture Notes in Computer Science.

[6] Gregory J. E. Rawlins and Derick Wood. On the optimal computation of finitely-oriented convex hulls. *Information and Computation*, 72:150–166, 1987.