A SIMPLE LINEAR ALGORITHM FOR COMPUTING EDGE-TO-EDGE VISIBILITY IN A POLYGON

by

Teren Gum

School of Computer Science
McGill University

January 1986

A dissertation
Submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

ISBN 0-315-34422-9

Abstract

One of the most recurring themes in many computer applications such as graphics, automated cartography, image processing and robotics is the notion of visibility. We are concerned with the visibility between two edges of a simple n-vertex polygon. Four natural definitions of edge-to-edge visibility are proposed. There exist $O(n \log n)$ algorithms and complicated $O(n)$ algorithms to solve this problem partially and indirectly. A simple, efficient, direct, linear running time, and thus optimal algorithm is presented to determine edge-to-edge visibility under any of the four definitions. The algorithm also identifies the visibility region if it exists.

Rèsumè

L'un des thèmes les plus récurrents dans de nombreux domaines d'application informatique tels le graphisme, la cartographie automatisée, le traitement d'images et la robotique est la notion de visibilité. Nous considérons le problème de la visibilité entre deux côtés dans un polygône simple à n points. Quatre définitions naturelles de visibilité côté-à-côté sont proposées. Il existe des algorithmes $O(n \log n)$ et des algorithmes $O(n)$ compliqués pour résoudre ce problème de façon partielle et indirecte. Nous présentons un algorithme simple, efficace, direct et linéaire en temps d'exécution pour déterminer la visibilité côté-à-côté avec n'importe des quatres définitions. L'algorithme identifie également la région de visibilité si ce dernier existe.

To My

Wonderful and Beloved

Mother

Acknowledgements

I would like to extend my deepest gratitude and thanks to my thesis supervisor, Prof. Godfried T. Toussaint, for suggesting the interesting topic, for his continuous advice, enthusiastic guidance and consistent support, and for creating a free atmosphere for research during the preparation of this thesis.

I am grateful to Prof. David Avis for his many useful suggestions and constructive criticism.

Special thanks go to fellow students Mary McQueen and David Rappaport for their helpful comments and careful readings of the manuscript.

I want to thank Miss Shui-Kit Wong for her constant encouragement and invaluable love, particularly during the course of this work, and for her help in drawing all the figures.

Since I began my study in Canada, it has been my blessing to know many best friends, especially Wai-Ling Kwok, Chui-Hung Chan, Shing-Tung Chan, Yu Mong, and Tak-Yin Wong, who have directly or indirectly contributed to the accomplishment of this work in diverse ways.

I also appreciate the financial support from the School of Computer Science, McGill University through teaching assistantships and the Natural Sciences and Engineering Research Council of Canada (from G.T. Toussaint's grant No. A-9293).

Table of Contents

CHAPTER ONE
Introduction

The notion of visibility or hidden line problem in geometric objects is one that appears in many applications: the central problem of computer graphics [Fr&Lo], image processing [Da&Be], and surveillance and control of robots [Nl]. A reasonable amount of research work has been devoted to solving this problem in two and three dimensional cases [Gr]. Several papers [Da&Be], [Sm], [Le&Pr1], [El&Av], [Le], [El&Av&To], [Ml], [To3], [Ra&To] have appeared concerning the problem of visibility in a polygonal region from a fixed point.

Let $P=(p_1, p_2, ..., p_n)$ be a *simple planar polygon*. Indices are taken modulo n throughout this thesis. We assume the polygon is in *standard* form, that is, all vertices are distinct, no three consecutive vertices are co-linear and vertices are given in clockwise order so that the interior of the polygon lies to the right as the edges are traversed. We say that a line segment lies *inside* P if it does not intersect the exterior of P. Two points are said to be *visible* if the line segment joining them lies inside P. The *vertex visibility polygon*, or just *visibility polygon*, *of* P *from a point* x in the plane, is that region of P *visible* from x. It has applications in path planning [Do] [Lo&We] and separability problems [Sc&To] [To&El] [To&Sc] in robotics, pattern recognition [Da&Be], and computer generated pictures of three dimensional objects [Cu&Le] [Ra] [Ya]. Shamos [Sm] presented a linear time algorithm for computing the visibility polygon but counterexamples are shown in [El1]. Freeman & Loutrel [Fr&Lo] and Davis & Benedikt [Da&Be] have proposed $O(n^2)$ and at least $O(n \log n)$ time algorithms for this problem respectively. ElGindy & Avis [El&Av] and D. T. Lee [Le] solve the same problem optimally in linear time. Melkman [Ml] improves the algorithm of [Le]. Asano [As] has an algorithm that can handle polygons with "holes".

The *vertex-visibility graph*, or the *viewability graph* [Sm] is a graph which has applications in computer vision [Sp&Ha1], shape decomposition [Sp&Ha2] and classification of planar shapes in pattern recognition [Av&El]. It is a graph whose nodes correspond to the vertices of the polygon and in which two nodes are adjacent if the associated vertices are visible. By computing the vertex visibility polygons for every vertex, the *vertex-visibility graph* can be computed in $O(n^2)$ time [Av&El].

A topic which has not been as much investigated as visibility from a *point* concerns the notion of *visibility from an edge*. A polygon P is *weakly visible from an edge uv* if for every point *w* in P, there exists a point *z* on edge *uv* such that edge *wz* lies inside P. Given a polygon P and a specified edge *uv* of P, the *weak edge visibility polygon*, or just *edge visibility polygon*, of P from edge *uv* is the region of P that sees at least one point of edge *uv*. Intuitively, it is the region of P *visible*, at one time or another, by a guard patrolling edge *uv*. The weak edge visibility polygon solves the *k-reachability* problem in $O(n \log n)$ time [El2] which has applications in planar separability in robotics, computing the reachable work area of a robot arm, and printed circuit routing [Hi]. ElGindy [El1] presented an $O(n^2)$ algorithm to find the weak edge visibility polygon using vertex visibility polygons [El1]. Recently, ElGindy [El2], Lee & Lin [Le&Li], and Chazelle & Guibas [Ca&Gu] all independently proposed three different algorithms for computing the weak edge visibility polygon in $O(n \log n)$ time. More recently, Toussaint [To1] reduced the complexity to $O(n)$. In the case where the polygon may have n "holes", Suri & O'Rourke [Su&OR] present an $O(n^4)$ algorithm for computing the boundary of the polygonal region *visible from an edge* and prove that it is optimal.

In this thesis we consider the problem of computing *edge-to-edge visibility* in a simple polygon. Given a polygon P and a pair of edges *uv* and *xy*, it is desired

to determine whether edge uv and xy are *visible* and. if so, report the *visible* part of both edges. There appear to be four natural definitions of visibility between a pair of edges:

a) Edge uv is said to be *completely visible* from edge xy if for all points z on edge xy and all points w on edge uv, w and z are *visible*.

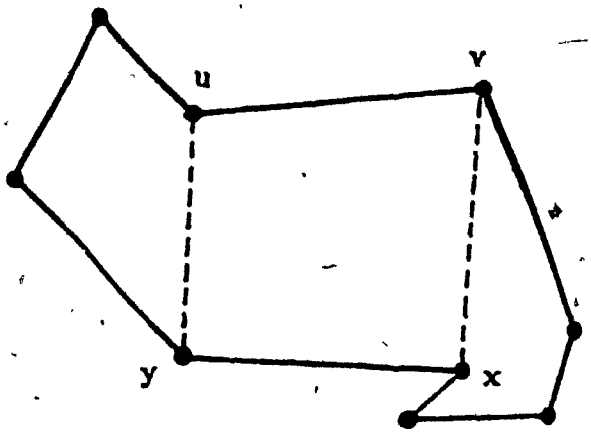
b) Edge uv is said to be *strongly visible* from edge xy if there exists a point z on edge xy such that for all points w on edge uv, w and z are *visible*.

c) Edge uv is said to be *weakly visible* from edge xy if for each w on edge uv, there exists a point z on edge xy such that w and z are *visible*.

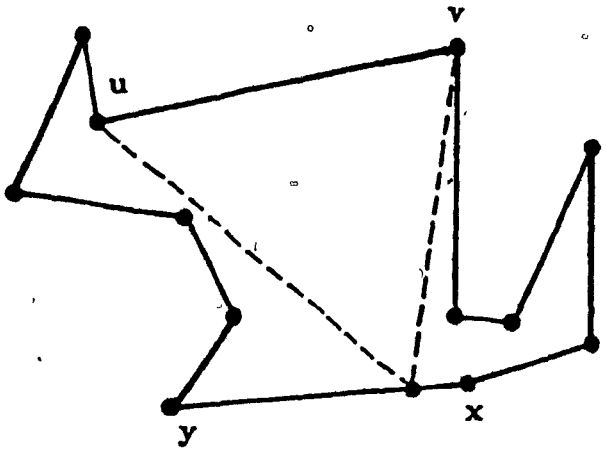
d) Edge uv is said to be *partially visible* from edge xy if there exists a point w on edge uv and a point z on edge xy such that w and z are *visible*.

These definitions are illustrated in Figure 1. As a motivation for the definitions, consider the problem of placing a watchman on edge xy so that he can observe edge uv. If he can be placed anywhere on edge xy and still be able to observe all of edge uv, we will have *complete visibility*. If there exists some fixed location on edge xy from where he can observe all of edge uv, we will have *strong visibility*. With only *weak visibility*, however, the guard must patrol up and down part of edge xy in order to observe all of edge uv. Finally, with *partial visibility*, the watchman can only observe part of edge uv, even if he patrols the length of edge xy.

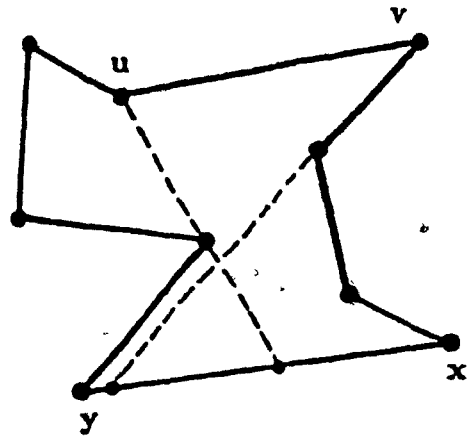
A graph analogous to the *vertex-visibility graph* for a polygon P can be defined in terms of edges. The *edge visibility graph* is a graph whose nodes correspond to the edges of the polygon and in which node n_1 is adjacent to node n_2 if the edge associated with node n_1 is visible from the edge associated with node n_2. So, we can define four kinds of edge visibility graphs: complete, strong, weak, and partial, under the four corresponding edge-to-edge visibilities. The complete and partial edge visibility graphs are undirected since complete and par-



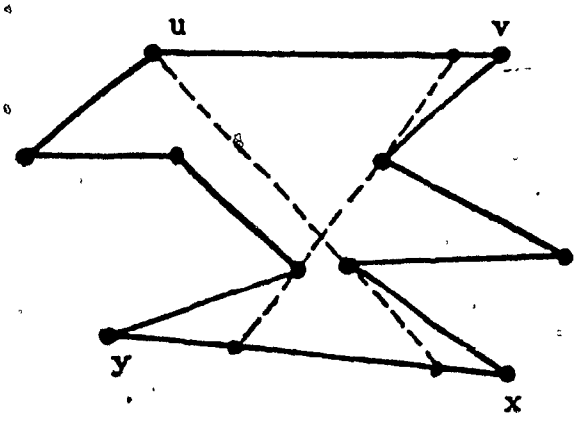
a. complete visibility



b. strong visibility



c. weak visibility



d. partial visibility

FIGURE 1 : Visibilities between two fixed edges

tial edge-to-edge visibility are symmetric, while the strong and weak edge visibility graphs are directed since strong and weak edge-to-edge visibility is not symmetric.

There are two methods in the literature to partially solve the edge-to-edge visibility problem. One way of doing this is to compute the weak edge visibility polygon of P from one edge, say edge uv, with any of the algorithms in [El2], [Le&Ll], [Ca&Gu], or [To1] and subsequently check whether the other edge intersects an edge of the weak edge visibility polygon. This approach yields an algorithm for the *weak and partial edge-to-edge visibility* problem. In [To2] Toussaint shows that the shortest path solves the partial edge-to-edge visibility problem and with a triangulation of P as a preprocessing step, a *partial edge-to-edge visibility* query in P can be answered in $O(n)$ time. Both approaches run in $O(n \log n)$ time and achieve $O(n)$ time with the fairly involved and complicated linear time triangulation algorithm recently discovered by Tarjan & Van Wyk [Ta&Va]. The details will be discussed in chapter two.

In chapter three, we exhibit a simple, efficient, direct and linear time algorithm for determining *edge-to-edge visibility* and reporting the *visible* part of both edges under any of the four definitions.

The last chapter discusses some related open problems, and suggests some directions for further research.

CHAPTER TWO
Previous Methods

Within the short literature of computational geometry, two other different problems, determining the weak edge visibility polygon and the shortest path between two vertices of a simple polygon, can be used indirectly to partially solve the edge-to-edge visibility problem.

2.1 Edge-to-Edge Visibility via Weak Edge Visibility Polygon

Avis & Toussaint [Av&To] considered what might be termed the "jail-house" problem, that is, checking the weak visibility of a simple polygon from an edge. It is motivated in [Av&To] by checking if the interior of the polygon can be watched by a mobile guard patrolling an edge of the polygon. They introduced three notions of polygonal visibility from an edge in a simple polygon: complete, strong and weak visibility, similar to the edge-to-edge visibility definitions in chapter one, and then presented a linear running time algorithm to solve this problem under any of the three visibilities. A more difficult problem is determining the visible region of a simple polygon from a given edge, or the edge visibility polygon, under the three visibilities. Recently, the problem of finding the *weak* edge visibility polygon received a lot of attention. Five different algorithms have been developed.

Ghosh & Shyamasundar [Gh&Sy] claimed an $O(n)$ algorithm. The main idea of their algorithm is that the weak edge visibility polygon is the union of the vertex visibility polygons from a few vertices, namely the two endpoints of the edge and those vertices that are visible from either one of them. Unfortunately, the algorithm does not always work as a counterexample is given in [El2].

Following the steps of the algorithm for checking the edge visibility of a simple polygon from an edge in [Av&To], Lee & Lin [Le&Li] proposed an $O(n \log n)$ algorithm for computing the weak edge visibility polygon. In their algorithm, after triangulating the polygon, vertices of the polygon are scanned twice to compute the furthest clockwise visible point on the edge, called the left intercept, and the furthest counterclockwise visible point on the edge, called the right intercept, and maintain the perspective blocking vertex, called the anchor point, in a concatenable queue associated with the vertices. Then the weakly visible region is determined from the relative positions of the left and right intercepts on the edge.

Recently, ElGindy [El2] offered an efficient method to compute the weak edge visibility polygon using a hierarchical representation of the polygon based on a decomposition into a set of simpler components: monotone polygons. The algorithm preprocesses the polygon in three passes. First, it cuts off parts of the polygon which cannot be weakly visible from the edge. The second pass performs a regular decomposition on P, that is, the polygon is decomposed into a set of components that are monotone with respect to a vertical line [Le&Pr3]. Finally, it computes the hierarchical representation of P based on the regular decomposition. The algorithm traverses the hierarchy and computes the left and right intercepts of the weakly visible vertices of the monotone components encountered on the edge, which takes up $O(n \log n)$ time. Unlike the previously described algorithm, this approach immediately determines the weak visibility of a vertex when the vertex is encounted during the scanning step, and this scanning process is terminated when it is clear that the remaining vertices are not weakly visible. Finally, it traverses the weakly visible vertices and computes the weakly visible parts of those regions bounded by the non-visible vertices, called the gap filter in [El2]. The entire process runs in $O(n \log n)$ time. An $O(n)$ algorithm for mono-

tone polygons is also shown in [El2].

More recently, Chazelle & Guibas [Ca&Gu] developed a different $O(n \log n)$ algorithm for finding the weak edge visibility polygon. Their main tool is the classic geometric duality in the two-sided plane introduced by the kinetic framework in [Gu&Ra&Si]. The weakly edge visible region in the plane dualizes to a double wedge in the dual plane. After triangulating the polygon as preprocessing, the algorithm partitions the double wedge in the dual plane into convex subdivisions based on the weakly visible region of the edges of the polygon in the plane. Then it computes the convex subdivision associated with the weakly visible region from the edge. This algorithm runs in $O(n \log n)$ time even after triangulation.

Most recently, Toussaint [To1] presented a linear time algorithm to determine the weak edge visibility polygon, which is also called the strong hidden-line problem in a simple polygon in [To1]. It combines results from visibility and shortest paths [Ca] with the linear time polygon triangulation algorithm discovered very recently by Tarjan & Van Wyk [Ta&Va]. The algorithm starts with the linear time triangulation [Ta&Va] on the polygon and the construction of the dual tree [Ca] also in linear time. Then it decomposes the polygon into *sleeves* corresponding to the branches in the dual tree. The main step is traversing the sleeves to cut off the invisible parts of each triangle traversed using the four shortest paths from each endpoint of the given edge to each endpoint of the triangulation diagonal. The algorithm first uses a procedure of ElGindy [El2] to compute all the shortest paths from the endpoints of the edge to all vertices of the polygon in linear time. This algorithm runs in linear time *even after* triangulation.

Given a simple polygon P and two edges uv and xy in P, the weak and partial edge-to-edge visibility problem can be solved in linear time once the two

weak edge visibility polygons from edge uv and edge xy are obtained. A subsequent check on whether the other edge intersects an edge of the weak edge visibility polygon suffices for partial edge-to-edge visibility. The weak edge-to-edge visibility is found if the weak edge visibility polygon contains the other edge. Using the last algorithm to obtain the weak edge visibility polygon, both entire processes can be done in $O(n)$ time theoretically, but the linear time polygon triangulation algorithm is very involved and uses heavy data structures such as recursive finger search trees [Br&Ta] [Hu&Mh] [Ma&Sl] and circularly level-linked 2-4 trees [Ho&Mh&Rs&Ta]. Alternatively, using any of the other three $O(n \log n)$ algorithms to obtain the weak edge visibility polygon, particularly the one by ElGindy [El2], our problem of weak and partial edge-to-edge visibility can be solved practically and efficiently but in $O(n \log n)$ time. Comprehensive pseudo code is given in [El2].

2.2 Edge-to-Edge Visibility via Shortest Paths

Given a simple polygon P and two points w, z in P, the *shortest path* (or *geodesic path*), SP(w,z|P), between w and z in P is a polygonal path connecting w and z which lies entirely in P, and such that the sum of its euclidean edge-lengths is a minimum over all other internal paths. Geodesic paths find application in many areas such as image processing [La&Ma], operations research [Le&Pr2], collision avoidance problems in robotics [To4], computing the trajectory of a racing car [Ca], wire routing in integrated boards [Hl], and optimization on a control problem of Bellman [Do] [El2].

Recently, Chazelle [Ca] and Lee & Preparata [Le&Pr2] independently discovered the same $O(n \log n)$ algorithm for computing SP(w,z|P). Both of these algorithms first triangulate P and then find the shortest path in $O(n)$ time by checking every triangulation edge to report those edges intersecting the shortest path and proceeding to compute the shortest path. Employing the linear time

polygon triangulation algorithm by Tarjan & Van Wyk [Ta&Va], the running time of both algorithms becomes linear.

More recently, a different $O(n \log n)$ algorithm due to ElGindy [El2] first constructs the hierarchical description of P based on a decomposition into monotone components in $O(n \log n)$ time and then searches it to report the sequence of monotone components containing the shortest path, from which the shortest path can be constructed later in linear time. Compared to the previous algorithms [Ca] and [Le&Pr1], the number of edges checked when searching the hierarchical description is much smaller than the number of edges in a triangulation of the polygon.

Toussaint [To2] proved that the partial edge-to-edge visibility between two edges uv and xy in a simple polygon P depends *entirely* on the convexities and the intersection of the two shortest paths SP(y,u|P) and SP(v,x|P). Using either the shortest path algorithm by Lee & Preparata [Le&Pr2] or Chazelle [Ca], the partial visibility between any two edges of P can be determined in $O(n)$ time given any $O(n \log n)$ triangulation [Ca] [Ga&Jo&P1&Ta] [Pr&Sm] as a preprocessing step [To2]. Thus, the partial edge-to-edge visibility can be determined in $O(n \log n)$ time. Again, using the linear time triangulation of polygon, the complexity is reduced theoretically to $O(n)$ time, but the entire algorithm becomes very involved and impractical. Alternatively, using the shortest path algorithm by ElGindy [El2], the running time remains $O(n \log n)$ while the programming is much simplified. With some modifications similar to those methods in this thesis, the shortest paths can be used to determine in linear time the edge-to-edge visibility according to any of the four definitions.

2.3 Conclusion

The weak edge visibility polygon approach has too much overhead to solve our problem efficiently. All the computations of the double scans in [Le&Ll], the

regular decomposition and the hierarchical representation of the polygon in [El2], the planar subdivision in [Ca], the dual tree in [To2] and the triangulation of the polygon in [Le&Li], [Ca] and [To2] have no direct contribution toward solving our problem. In addition, all these computations are much more complicated than need be for our problem and solve only two of the four edge-to-edge visibility problems. The shortest path approach is closer to our problem than the weak edge visibility polygon approach, but it is still an indirect one. We will present a simple, efficient, direct and linear running time solution to the edge-to-edge visibility problem.

CHAPTER THREE
The Edge-to-Edge Visibility Algorithm

We start by defining some terminology used in this chapter in section one. Then some preliminary results are given in section two. Section three describes in detail a linear running time algorithm for determining the edge-to-edge visibility of a simple polygon given any two fixed edges under any of the four edge-to-edge visibilities defined in chapter one. We believe that the algorithm presented here is the simplest, most direct and efficient one.

3.1 Definitions and Terminology

The *right half plane* of an edge wz, RHP(w,z), is the half plane defined by the line passing through wz, and lying on the right of edge wz, including the line wz. The *half line* of an edge wz, HL(w,z), is the infinite half line starting at w and passing through z. Edge uv is *partially facing* edge xy if and only if one endpoint of edge xy is in RHP(u,v). Edge uv is *totally facing* edge xy if and only if x and y are in RHP(u,v). Edge uv is *facing* edge xy if and only if edge uv is partially or totally facing edge xy (see Figure 2).

A polygonal chain $c_1, c_2, \ldots, c_p$, denoted by $C(c_1,c_p)$, is said to have a *cut* through the quadrilateral $Q=(uvxy)$ if and only if there exists a sub-chain $c_i, \ldots, c_j$, where $1 \leq i < j \leq p$ with c_i and c_j lying outside quadrilateral Q such that

1) all points except c_i and c_j on the sub-chain lie in the interior of Q, and
2) the two intersections are on the two opposite edges vx and yu of Q respectively, that is, edge $c_i c_{i+1}$ intersects edge vx, and edge $c_{j-1}c_j$ intersects edge yu, or, edge $c_i c_{i+1}$ intersects edge yu, and edge $c_{j-1}c_j$ intersects edge vx (see Figure 3).

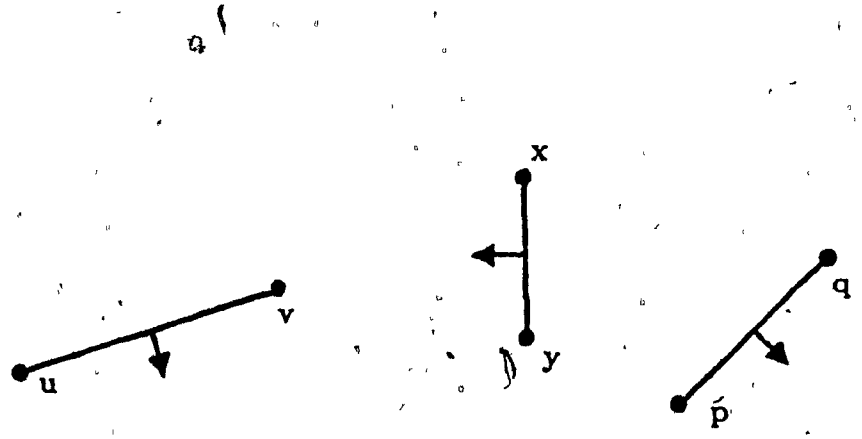
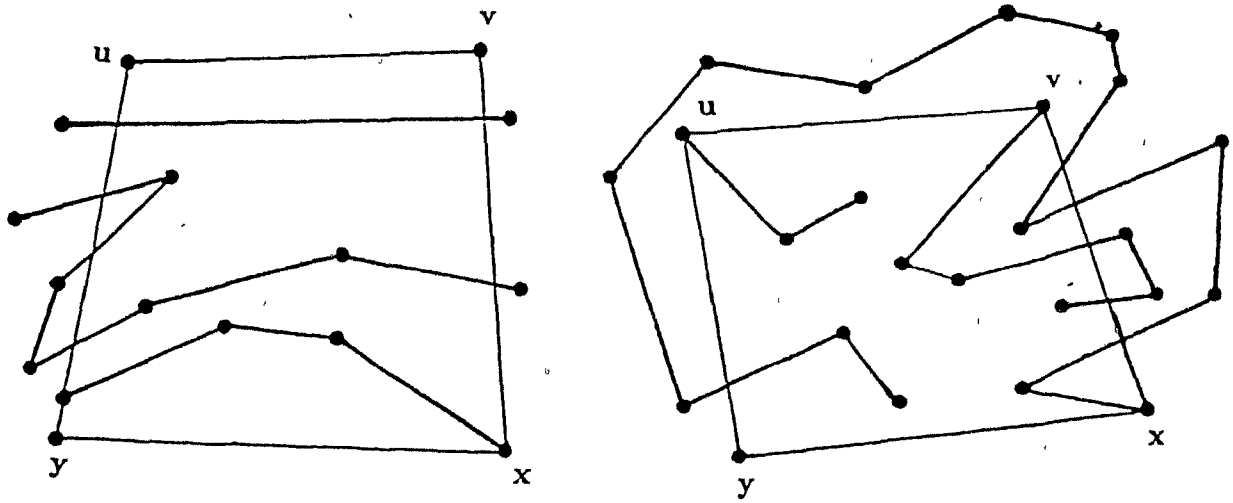


FIGURE 2 : Examples of facing. uv is partially facing xy, xy is totally facing uv, and pq is not facing uv



a. Chain that cut through the Quadrilateral uvxy

b. Chain that do not cut through the Quadrilateral uvxy

FIGURE 3 : Cuts

Let $P=(p_1, p_2, ..., p_n)$ be a simple polygon in the plane. Consider all the intersection points between edge pq and $C(p,q)$. An intersection point p_i is said to be *going out* if p_{i-1} is on the left of edge pq. An intersection point p_i is said to be *going in* if p_{i-1} is on the right of edge pq. Consider the intersection points on edge pq are sorted such that $p = p_{I_1}, p_{I_2}, ..., p_{I_t} = q$ is monotonic on edge pq. The *reduced chain* of $C(p,q)$, $RC(p,q)$, is edge $p_{I_1}p_{I_2}$, $C(p_{I_2},p_{I_3})$, edge $p_{I_3}p_{I_4}$, $C(p_{I_4},p_{I_5})$, ..., edge $p_{I_{t-2}}p_{I_{t-1}}$, $C(p_{I_{t-1}},p_{I_t})$ (if p_{I_t} is *going out*), or, $C(p_{I_1},p_{I_2})$, edge $p_{I_2}p_{I_3}$, $C(p_{I_3},p_{I_4})$, edge $p_{I_4}p_{I_5}$, ..., $C(p_{I_{t-2}},p_{I_{t-1}})$, edge $p_{I_{t-1}}p_{I_t}$ (if p_{I_t} is *going in*). Note that if p_{I_i} is *going out*, $p_{I_{i+1}}$ is *going in* and if p_{I_i} is *going in*, $p_{I_{i+1}}$ is *going out* (see Figure 4). Given two edges uv and xy of a simple polygon P, the *inner convex hull* of the reduced chain yu, $ICH(y,u)$, is the convex hull of all the points on $C(y,u)$ lying in Quadrilateral($uvxy$) except edge yu (see Figure 5).

3.2 Preliminary Results

We begin with some observations about visibility between edges. First, the edge-to-edge visibility definitions mean that partial visibility implies weak visibility, implies strong visibility, implies complete visibility, but not the converse (see Figure 1). Secondly, the definition of completely visible is *symmetric*, that is, edge uv is completely visible from edge xy if and only if edge xy is completely visible from edge uv. Partial visibility is also *symmetric*. Thirdly, strong visibility and weak visibility are not *symmetric* as illustrated by the examples in Figure 1(b) and 1(c). Fourthly, edge uv is completely visible from edge xy if and only if u and v are visible from y and x respectively. Finally, we note that a simple polygon is convex if and only if every pair of edges is completely visible.

Lemma 1 : Given any two edges uv and xy of a simple polygon P, edge uv is either totally facing, partially facing or not facing edge xy, and vice versa. Also edge uv and edge xy cannot partially face each other (Table 1).

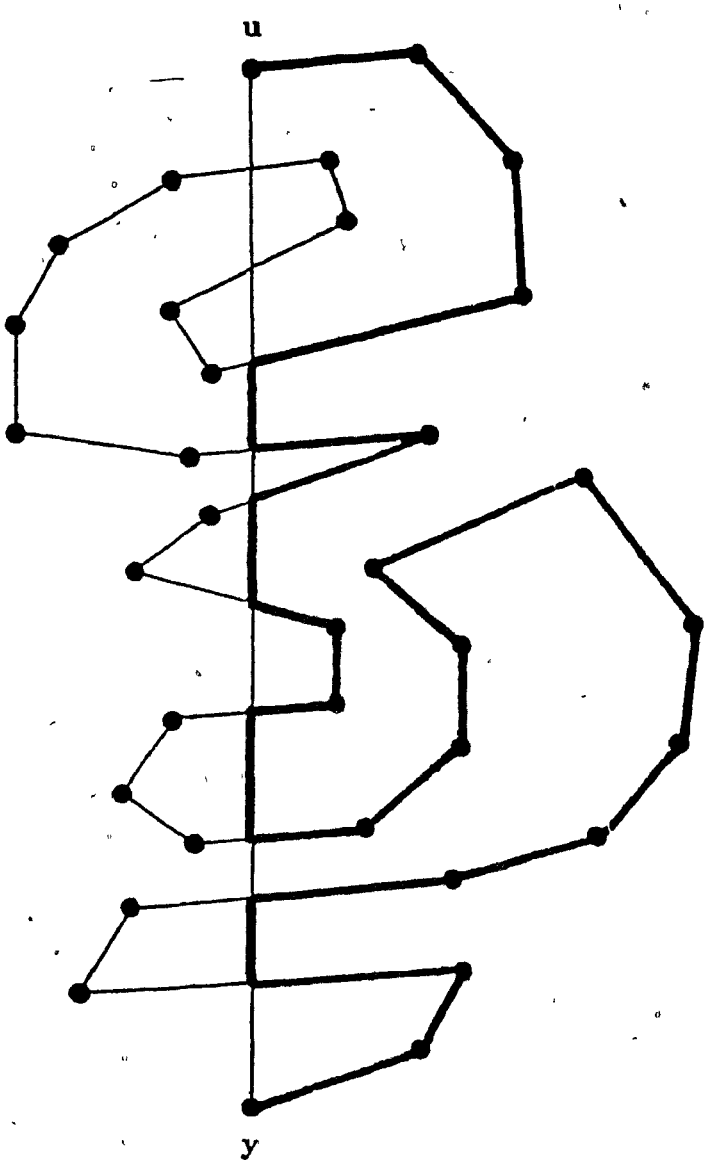


FIGURE 4 : Reduced Chain RC(y,u)

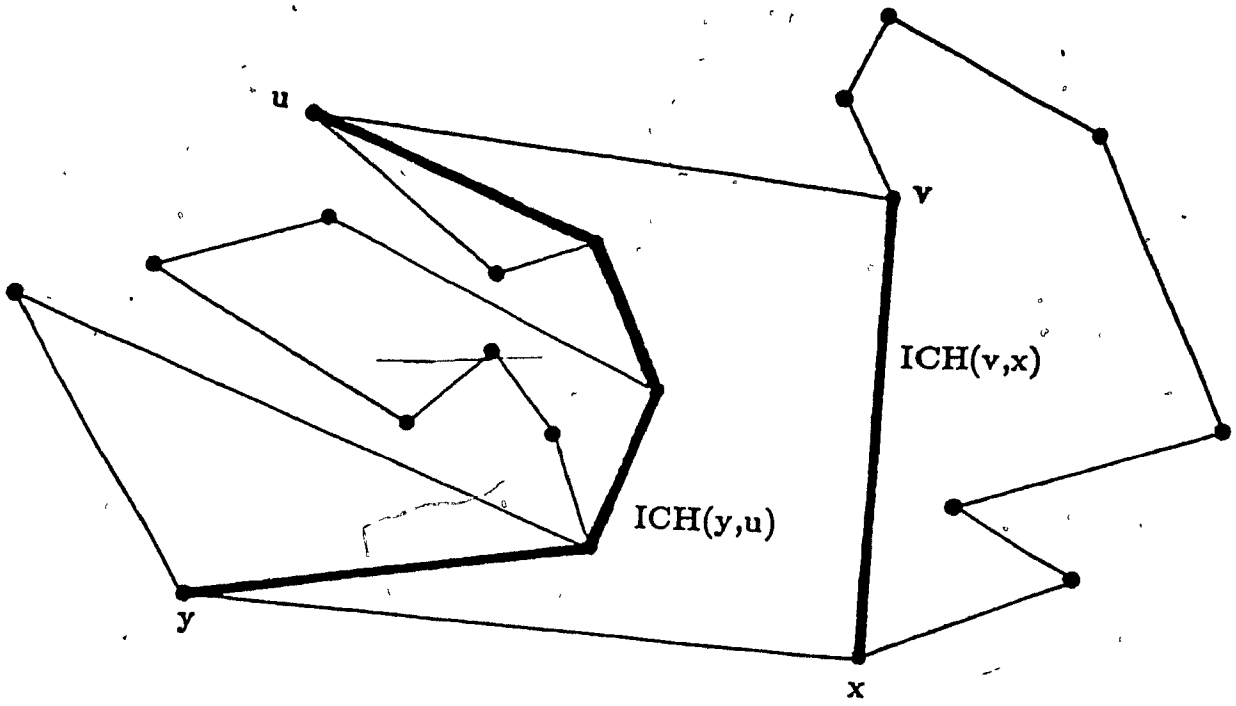


FIGURE 5 : Inner Convex Hull

Proof : The first part follows straightforwardly from the definitions: either both x and y are in RHP(u,v), or either x or y is in RHP(u,v), or both x and y are not in RHP(u,v). Secondly, assume edge uv and edge xy are partially facing each other. Consider the intersection point, o, of line xy and line uv. Since u or v exclusively is in RHP(x,y), o must lie between u and v on the line uv. And since x or y exclusively is in RHP(u,v), o must lie between x and y on the line xy. Therefore, edge uv intersects edge xy at o, which violates the simplicity of the polygon. ■

	xy is not facing uv	xy is partially facing uv	xy is totally facing uv
uv is not facing xy	case 1	case 2	case 3
uv is partially facing xy	case 4	no such case	case 5
uv is totally facing xy	case 6	case 7	case 8

Table 1 : Eight Canonical Positions of Two Fixed Edges

Lemma 2 :

a) If both edges are not facing each other, then only vertex-to-vertex visibility is possible. (case 1 in Table 2)

b) If one of the two edges is not facing the other edge while the other does, then only vertex-to-vertex or vertex-to-edge visibility are possible. (case 2, 3, 4, and 6 in Table 2)

Proof : The visible region for any point w on edge uv, except the two endpoints, is RHP(u,v). Let w be any point in the interior of edge uv, and z be any point in the interior of edge xy.

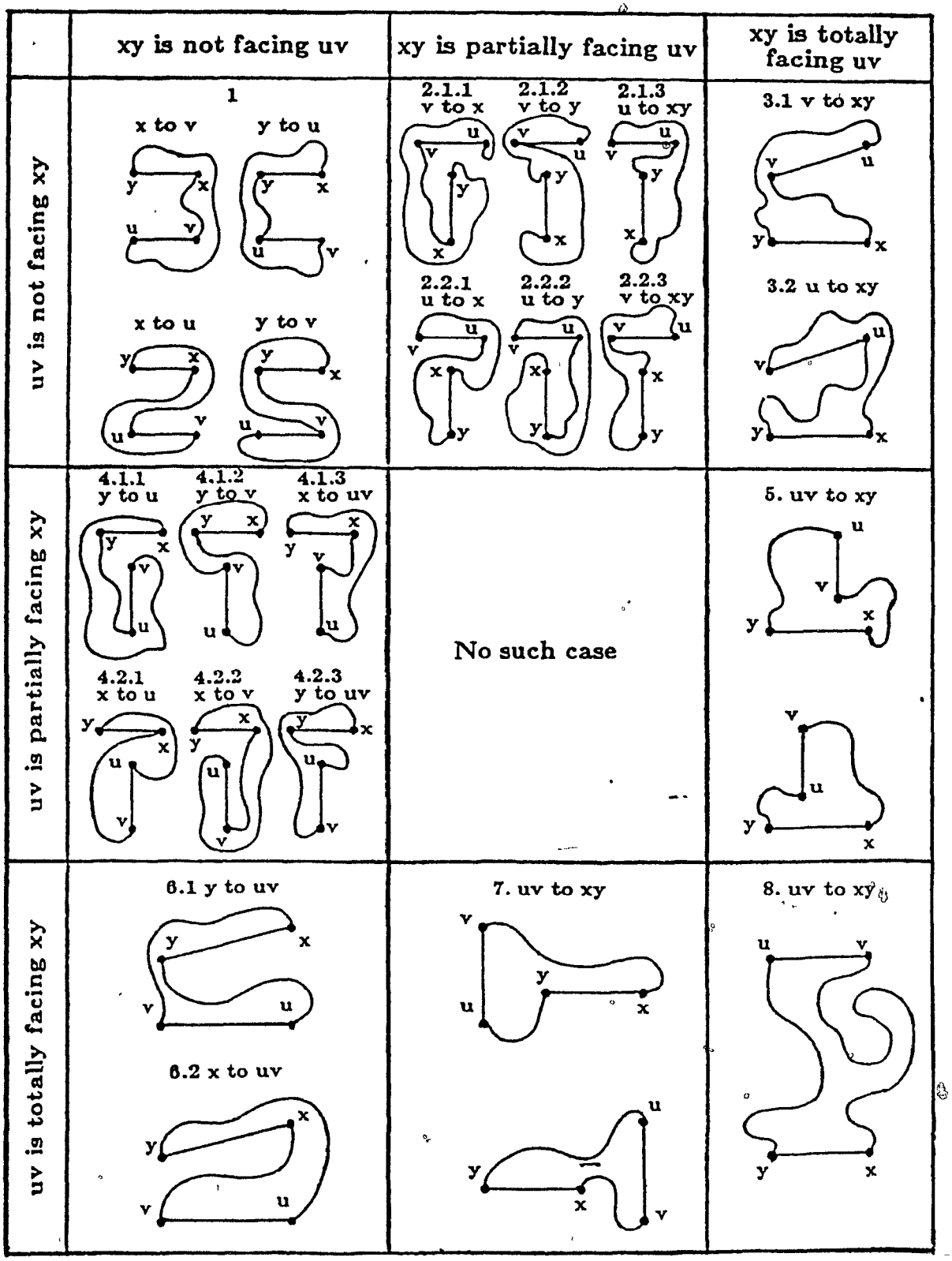


Table 2 : Illustration of the eight cases of two fixed edges

a) If both edges are not facing each other, then x and y are not in RHP(u,v), so edge xy is not visible from w; also, u and v are not in RHP(x,y), so edge uv is not visible from z. Therefore, only visibilities between the four endpoints are possible.

b) If edge xy is not facing edge uv, then u and v are not in RHP(x,y), hence, edge uv is not visible from z. Therefore, only visibility between vertex x or y and edge uv is possible. ■

Given two fixed edges uv and xy of a simple polygon, from lemmas 1 and 2 it follows that we have eight canonical cases of the general edge-to-edge visibility problem as shown in Table 1 and 2.

The different situations that arise are shown in Table 2. In case 2, edge xy can only be visible from vertex u or v exclusively. We have two subcases, as edge xy is partially facing edge uv, either u or v exclusively is in RHP(x,y). If u is in RHP(x,y) (case 2.1), it is obvious that when v and x are visible (case 2.1.1), and when v and y are visible (case 2.1.2), no other types of visibility are possible. However, when neither x or y is visible from v, we must consider the visibility from u to edge xy (case 2.1.3). A parallel situation holds if v is in RHP(x,y) (case 2.2.1 to 2.2.3). In case 3, when edge xy is totally facing edge uv, either u or v exclusively can be visible from edge xy (case 3.1 and 3.2). Cases 4 and 6 are symmetrical to cases 2 and 3, respectively. In cases 5 and 7, one edge is partially facing and is totally faced by the other edge. We will call these cases *improper edge-to-edge visibility* as different from *proper edge-to-edge visibility* (case 8).

Now, we have four subproblems: vertex-to-vertex visibility, vertex-to-edge visibility, improper edge-to-edge visibility and proper edge-to-edge visibility. These subproblem are summarized in Table 3. As a preliminary step in the general algorithm, we can determine in constant time which of the above cases holds for a given query.

	xy is not facing uv	xy is partially facing uv	xy is totally facing uv

uv is not facing xy	**case 1**
	1. **if** x to v is visible **then** partial visibility;
	2. **if** x to u is visible **then** partial visibility;
	3. **if** y to v is visible **then** partial visibility;
	4. **if** y to u is visible **then** partial visibility;
	5. no visibility.
		case 2
		if u is in RHP(x,y) **then**
		1. **if** v to x is visible **then** partial visibility;
		2. **if** v to y is visible **then** partial visibility;
		3. u to xy visibility
		else
		1. **if** u to x is visible **then** partial visibility;
		2. **if** u to y is visible **then** partial visibility;
		3. v to xy visibility.
			case 3
			v to xy visibility;
			if no visibility **then** u to xy visibility.

uv is partially facing xy	**case 4**
	if x is in RHP(u,v) **then**
	1. if y to u is visible **then** partial visibility;
	2. **if** y to v is visible **then** partial visibility;
	3. x to uv visibility
	else
	1. **if** x to u is visible **then** partial visibility;
	2. **if** x to v is visible **then** partial visibility;
	3. y to uv visibility.
		no such case	**case 5**
			Improper uv to xy visibility.

uv is totally facing xy	**case 6** y to uv visibility; **if** no visibility **then** x to uv visibility.	**case 7** Improper xy to uv visibility	**case 8** uv to xy visibility

Table 3 : Preliminary Step of Finding the Type of Edge-to-Edge Visibility

3.3 The Algorithm

In this section, we present a linear running time algorithm to solve the edge-to-edge visibility problem. We assume we have done the preliminary step discussed in last section in order to solve the general edge-to-edge visibility problem.

3.31 Edge-to-Edge Visibility

First, we consider the proper edge-to-edge visibility subproblem. The algorithm will be shown and a proof of correctness will follow. The basic idea is to find the two inner convex hulls of $C(y,u)$ and $C(v,x)$, respectively, bounded at the endpoints, and then to determine the visibility by analyzing these two chains.

ALGORITHM EDGE-TO-EDGE VISIBILITY

Input : A simple polygon $P=(p_1, p_2, \ldots, p_n)$, and two edges in P, uv and xy, totally facing each other.

Output : the visibility between edge uv and edge xy.

Begin

1. [*initialization*] Build a double circular linked list for the vertices of the simple polygon.
2. [*in case of a cut*] **If** $C(v,x)$ *or* $C(y,u)$ *cut* through the Quadrilateral $Q(uvxy)$, **then** return with 'No Visibility'.
3. [*locate the intersection points*] FIND intersection points $p_{I_1}=y, p_{I_2}, \ldots, p_{I_t}=u$ between $C(y,u)$ and edge yu, and $p_{I_{t+1}}=v, p_{I_{t+2}}, \ldots, p_{I_{t+l}}=x$ between $C(v,x)$ and edge vx, INSERT p_{I_i} between p_j and p_{j+1} into the linked list, where edge $p_j\ p_{j+1}$ intersects edge yu or edge vx at p_{I_i}, for $i=1,2,\ldots,l$. All inserted vertices are labeled as intersection points and are linked in a double circular list. We denote a point *going out* as $p_{I_{out}}$ and a point *going in* as $p_{I_{in}}$. If p_{I_1} or $p_{I_{t+1}}$ is *going out*, insert y or v respectively before it and consider it as *going in*. If p_{I_t} or p_{I_l} is *going in*, insert u or x respectively before it and consider it as *going out*.

4. [*find the reduced chain of* C(y,u)]
   For (*in* ,*out*) = (1,2), (3,4), ..., (k −1,k) **do**
   If $y\ p_{I_{in}}\ p_{I_{out}}\ u$ is monotonic
   then [*scan forward and delete*]
   a. scan forward starting from $p_{I_{out}}$ to the first intersection point $p_{I_{in}*}$ which is on edge $p_{I_{out}}\ u$;
   b. delete all points between $p_{I_{out}}$ and $p_{I_{in}*}$, exclusive, if any.
   else [*scan backward and delete*]
   a. scan backward along the intersection points list, locate $p_{I_{out}'}$ and $p_{I_{in}'}$ such that $p_{I_{out}}$ is between $p_{I_{out}'}$ and $p_{I_{in}'}$ on edge yu;
   b. scan forward starting from $p_{I_{out}}$ to the first intersection point $p_{I_{in}*}$ which is on edge $p_{I_{out}'}\ p_{I_{out}}$;
   c. delete all points between $p_{I_{out}'}$ and $p_{I_{in}*}$, exclusive, if any.
5. [*find the reduced chain of* C(v,x)] Analogous to step 4 with C(v,x) replacing C(y,u), v replacing y and x replacing u.
6. [*compute the inner Convex Hull*] Find the inner Convex Hull of C(v,x), ICH(v,x). Find the inner Convex Hull of C(y,u), ICH(y,u). Let u' be the point preceding u in ICH(y,u). Let v' be the point after v in ICH(v,x). Let u'' be the intersection of line uu' and line xy. Let v'' be the intersection of line vv' and line xy. Similarly, we define x', y' x'' and y''.
7. [*construct the polygon* H] H ← edge uv, ICH(v,x), edge xy, ICH(y,u)
8. [*identify the visibility region of edge uv from edge xy*]
   If $x = v''$ and $y = u''$
   then edge uv is *completely visible* from edge xy;
   If $yu''v''v$ are monotonic on edge xy
   then edge uv is *strongly visible* from edge xy between u'' and v'';
   If u'' and v'' are on edge xy and $v'v''$ does not intersect ICH(y,u) and $u'u''$ does not intersect ICH(v,x)
   then edge uv is *weakly visible* from edge xy between u'' and v'';
   If H is a simple polygon
   then edge uv is *partially visible* from edge xy between L_{uv} and R_{uv} on edge uv and L_{xy} and R_{xy} on edge xy, where L_{uv}, R_{uv}, L_{xy}, and R_{xy} are the intersections of edge uv and xy with the two critical lines of support L and R between two convex polygons constructed by ICH(v,x) and edge vx and by ICH(y,u) and edge yu.
9. [*identify the visibility region of edge xy from edge uv*] Analogous to step 8 with x replacing u and v replacing y.

End

Lemma 3 : Both C(v,x) and C(y,u) do not *cut* through Q($uvxy$) if and only if for all w on edge uv and all z on edge xy, there exists a polygonal path wz lies in the interior of the Q($uvxy$), except w and z, such that it does not intersect C(v,x) or C(y,u).

Proof : [if] Assume C(v,x) *or* C(y,u) has a *cut* through Q. Let r and s be the two vertices that realize the *cut*, that is, r is not s and all points in C(r,s), except r and s, lie in the interior of Q. Therefore, C(r,s) partitions Q into two regions at edge vx and edge yu. Obviously, for all w on edge uv and all z on edge xy, any path wz in the interior of Q will intersect C(r,s) (refer to Figure 6).

[only if] Assume C(v,x) and C(y,u) do not *cut* through Q($uvxy$). By definition, for all r and s on C(v,x) or on C(y,u), where $r < s$ and both lie outside Q($uvxy$), either, there is a point t between r and s that lies outside Q, or, if not, both of the two intersections of C(r,s) and Q lie on one of the edges vx or yu. In the former case, $r<t<s$, all three vertices r, t, s lie outside Q, and both C(r,t) and C(t,s) satisfy one of the two above cases. In the latter case, C(r,s) partitions Q into two polygons. Since the two points of intersection lie on the same edge, edge uv and edge xy will be on the same polygon. ■

Lemma 4 : In step 4 of ALGORITHM EDGE-TO-EDGE VISIBILITY,

after each iteration

a) all the visited intersection points are monotonic on edge yu;

b) all chains between each visited intersection pair, whose first point is a point on C(y,u) going out of Q($uvxy$) and whose second point is the one going in, lie inside Q($uvxy$);

c) C(y,$p_{I_{in}*}$) is simple;

when the algorithm terminates

d) all points of P lying outside Q($uvxy$) are deleted;

e) C(y,u) lies inside Q($uvxy$) except for the intersection points;

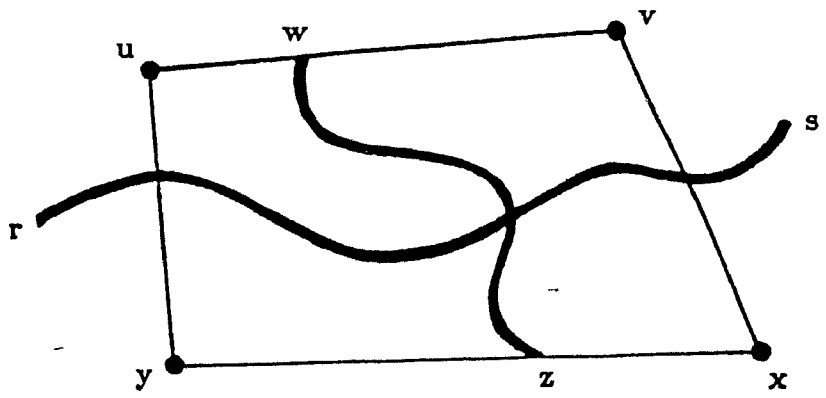


FIGURE 6

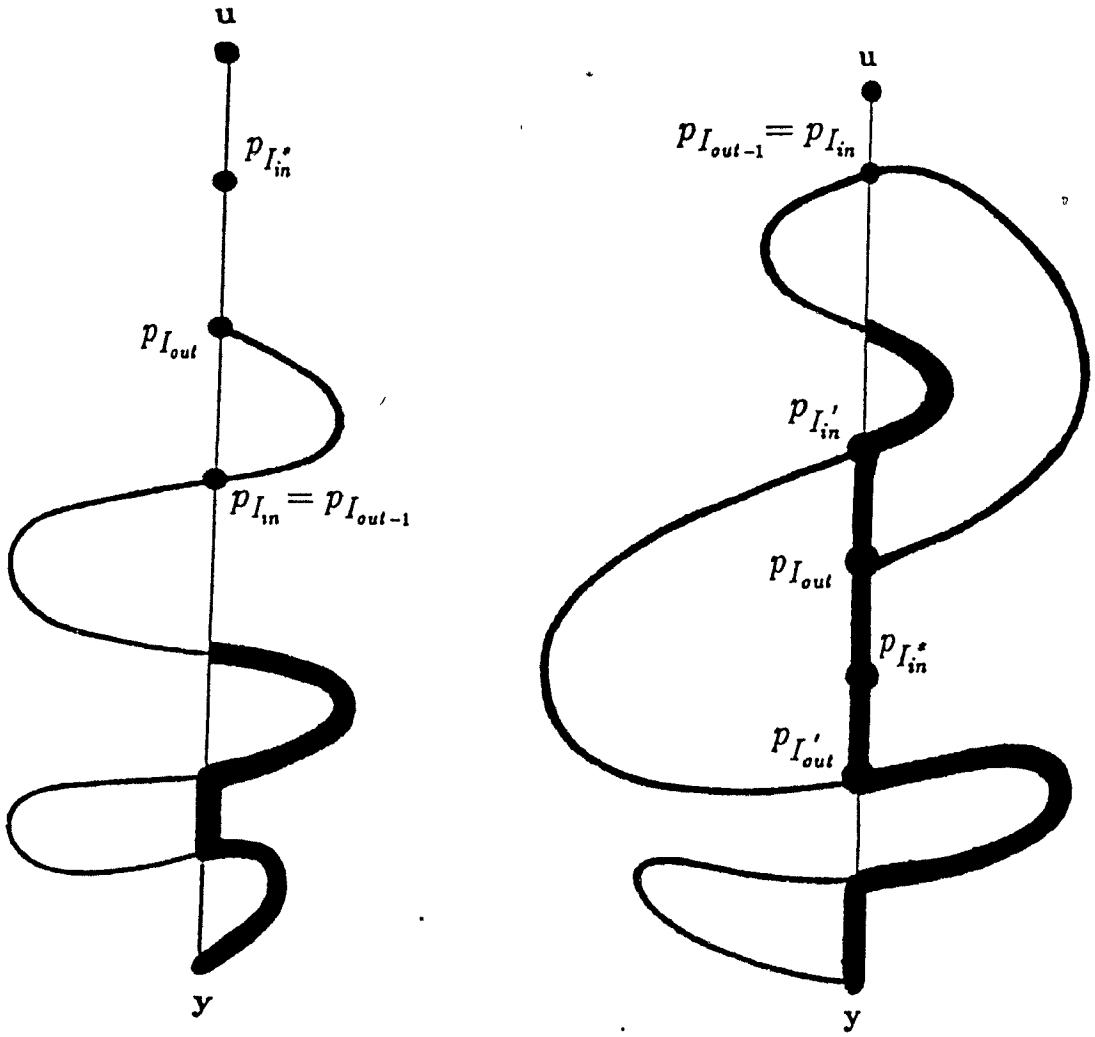


FIGURE 7

f) $C(y,u)$ is simple.

Proof :

a) (refer to Figure 7) If $p_{I_{out-1}}\ p_{I_{out}}\ u$ is monotonic, $p_{I_{in}*}$ is on edge $p_{I_{out}}u$, and $p_{I_{out}}\ p_{I_{in}*}\ u$ is monotonic, thus keeping all the visited intersection points monotonic. If not, then all the visited intersection points after $p_{I_{in}'}$ are on $C(p_{I_{in}'},p_{I_{out}})$ which is on $C(p_{I_{out}'},p_{I_{in}*})$, and hence they are deleted. In addition, $p_{I_{in}*}$ is on edge $p_{I_{out}'}\ p_{I_{out}}$, and $p_{I_{out}'-1}\ p_{I_{out}'}\ p_{I_{in}*}$ is monotonic, thus keeping all the visited intersection points monotonic.

b) If $p_{I_{out-1}}\ p_{I_{out}}\ u$ is monotonic, consider the last pair $p_{I_{out-1}}\ p_{I_{out-2}}$ and the current pair $p_{I_{out}}\ p_{I_{in}*}$ just added. $C(p_{I_{out-1}},u)$ has its first intersection with edge yu at $p_{I_{out-1}}$, so $C(p_{I_{out-1}},p_{I_{out}})$ intersects edge yu only at $p_{I_{out-1}}$ and $p_{I_{out}}$. $C(p_{I_{out-1}},p_{I_{out}})$ cannot intersect edge uv or edge xy since the polygon is simple. Furthermore, it cannot intersect edge vx or edge yu since otherwise it would have a *cut* through $Q(uvxy)$. Therefore, $C(p_{I_{out-1}},p_{I_{out}})$ lies completely inside $Q(uvxy)$ except for the two endpoints. If $p_{I_{out-1}}\ p_{I_{out}}\ u$ is not monotonic then deleting the pair $p_{I_{out-1}}\ p_{I_{out-2}}$ up to the pair $p_{I_{in}'}\ p_{I_{out}'}$ and adding the pair $p_{I_{out}'}\ p_{I_{in}*}$ has no effect on the chain between each visited intersection pair.

c) If $p_{I_{out-1}}\ p_{I_{out}}\ u$ is monotonic, from the previous step, $C(y,p_{I_{out-1}})$ is simple. From part b, $C(p_{I_{out-1}},p_{I_{out}})$ lies inside $Q(uvxy)$ except for its two endpoints, i.e., it does not intersect edge yu except for its two endpoints. $C(p_{I_{out-1}},p_{I_{out}})$ does not intersect the chains between intersection pairs, since otherwise the polygon is not simple. Since all chains between visited intersection pairs lie inside $Q(uvxy)$ (part b of Figure 7) and are monotonic on edge yu (part a of Figure 7), and all the intersection pairs are on edge yu, edge $p_{I_{out}}p_{I_{in}*}$ does not intersect $C(y,p_{I_{out}})$. Therefore, $C(y,p_{I_{in}*})$ is simple. If $p_{I_{out-1}}\ p_{I_{out}}\ u$ is not monotonic, from the previous step $C(y,p_{I_{out-1}})$ is simple and $C(y,p_{I_{in}'})$ is simple. Since $p_{I_{in}*}$ is on edge $p_{I_{out}'}\ p_{I_{in}'}$,

$C(y,p_{I_{in}*})$ is simple.

d) All points that are outside $Q(uvxy)$ are the points between a point, $p_{I_{out}}$, going out of Q, and a point $p_{I_{in}}$ going into Q. In each step the algorithm deletes all points between a $p_{I_{out}}$ point to a $p_{I_{in}}$ point. Consequently, all points outside Q will be deleted when the algorithm terminates.

e) From the results of part b and d, it is obvious that when the algorithm terminates, $C(y,u)$ lies outside $Q(uvxy)$ except for the intersection points.

f) From the results of part c, it is also obvious that when the algorithm terminates, $C(y,u)$ is simple. ■

Lemma 5.1 : Edge uv is *partially visible* from edge xy if and only if H is a simple polygon.

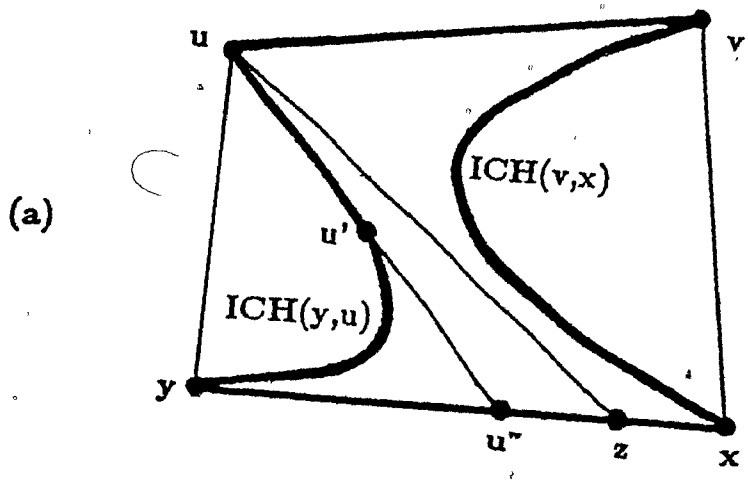
Proof : (refer to Figure 8) **[only if]** Let w on edge uv be visible from z on edge xy, then edge wz lies inside P, so edge wz does not intersect $C(v,x)$ and $C(y,u)$. Consequently, $ICH(v,x)$ and $ICH(y,u)$ do not intersect edge wz. Edge wz splits the $Q(uvxy)$ into two regions $R(uwzy)$ and $R(wvxz)$. Since $ICH(v,x)$ lies inside $Q(uvxy)$, it lies inside $R(wvxz)$; and since $ICH(y,u)$ lies inside $Q(uvxy)$, it lies inside $R(uwzy)$. Therefore, $ICH(v,x)$ does not intersect $ICH(y,u)$ and H is simple.

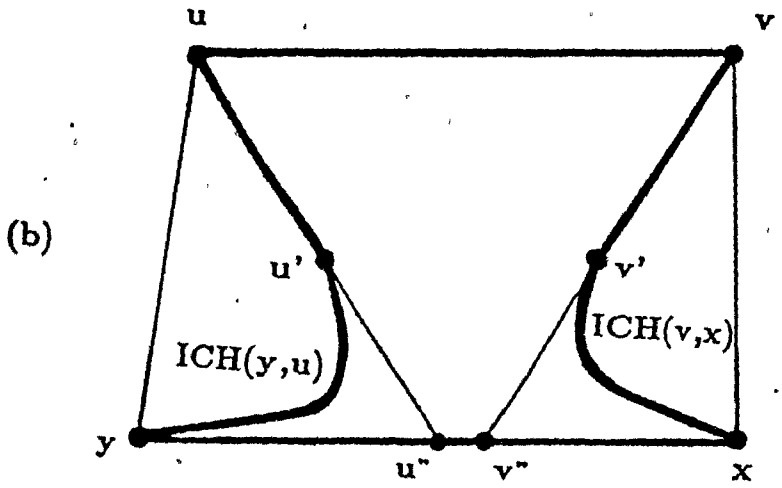
[if] If H is simple, ICH(y,u) does not intersect ICH(v,x). The polygon constructed by concatenating ICH(y,u) and edge yu, and the polygon constructed by concatenating ICH(v,x) and edge vx are both convex and do not intersect. By a well-known theorem in convexity [Se&Wl], two non-intersecting convex polygons are linearly separable, and by construction of H, the line separating the two polygons must intersect edge uv at some point w and edge xy at some point z. Since edge wz lies inside H, it lies inside P. Therefore, edge uv is partially visible from edge xy. ■

Lemma 5.2 : Edge uv is *weakly visible* from edge xy if and only if u'' and v'' are on edge xy and edge $v'v''$ does not intersect ICH(y,u) and edge $u'u''$ does not intersect ICH(v,x).

Proof : [only if] Suppose edge uv is weakly visible from edge xy, then there exists a point z on edge xy such that edge zu lies inside P (refer to Figure 9(a)). Neither ICH(v,x) nor ICH(y,u) can intersect edge zu. Edge zu splits the Q($uvxy$) into two regions, R(uzy) and R($uvxz$). Since ICH(v,x) lies inside Q($uvxy$), it lies inside R($uvxz$), and since ICH(y,u) lies inside Q($uvxy$), it lies inside R(uzy). Hence edge zu lies inside H and edge uu'' lies in R(uzy). Therefore, edge $u'u''$ cannot intersect ICH(v,x). Since ICH(y,u) is concave in H, edge uu'' cannot intersect it. Therefore edge uu'' lies inside H and u'' is on edge xy. An exactly parallel argument shows that edge $v'v''$ cannot intersect ICH(y,u), and v'' is on edge xy.

[if] There are two cases depending on whether or not edge uu'' intersect edge vv'' inside H. Suppose edge uu'' does not intersect edge vv'' inside H(refer to Figure 9(b)). Since u'' and v'' are on edge xy, edge $v'v''$ does not intersect ICH(y,u) and edge $u'u''$ does not intersect ICH(v,x), edge uu'' and edge vv'' split the Q($uvxy$) into three Regions R($uu''y$), R($uvv''u''$) and R(vxv''). Since ICH(y,u) is inside Q($uvxy$) and concave in H, it lies in R($uu''y$) and since ICH(v,x) is inside Q($uvxy$) and concave in H, it lies in R(vxv''). So R($uvv''u''$) is inside H and thus inside P.


(a)
u
v
ICH(v,x)
u'
ICH(y,u)
y
u"
z
x


(b)
u
v
u'
v'
ICH(y,u)
ICH(v,x)
y
u"
v"
x

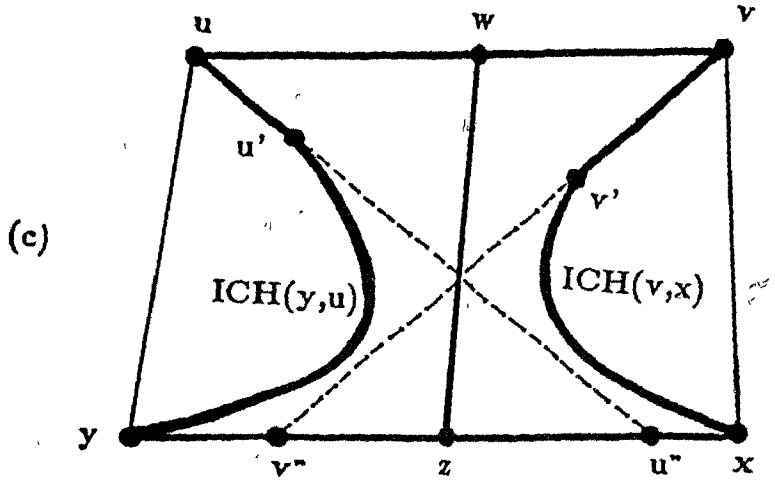

(c)
u
w
v
u'
v'
ICH(y,u)
ICH(v,x)
y
v"
z
u"
x

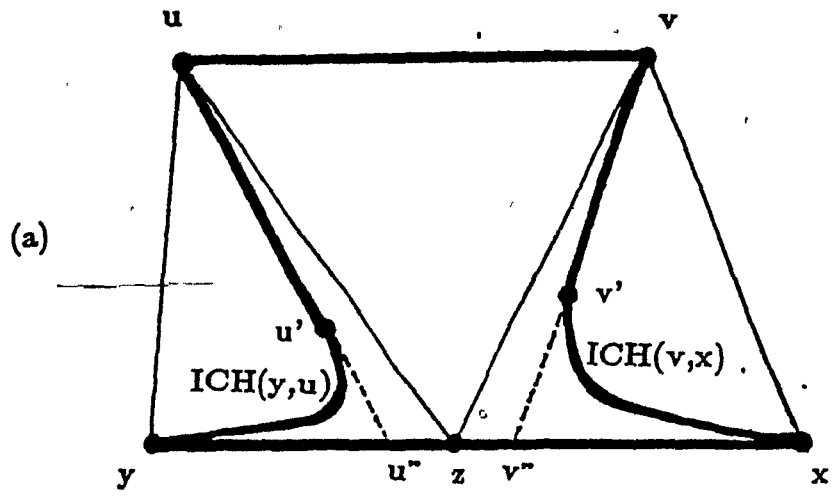
FIGURE 9

In this case, edge uv is strongly visible from edge $u"v"$. Therefore, edge uv is weakly visible from edge xy. On the other hand, suppose that edge $uu"$ and edge $vv"$ intersect at a point t inside H (refer to Figure 9(c)). ICH(y,u) does not intersect edge $u'u"$ because of its convexity and edge $vv"$ by assumption. ICH(v,x) does not intersect edge $v'v"$ because of its convexity and edge $u'u"$ by assumption. So edge $u'u"$ and edge $v'v"$ lie inside H. It follows that the two triangles utv and $u"tv"$ lie inside H and hence inside P. Choose any point w on edge uv and extend w through t to a point z on edge $u"v"$. Since edge wz lies inside the two triangles which are inside P, it follows that, any w on edge uv is visible from a point z on edge xy, and edge uv is weakly visible from edge xy. ■

Lemma 5.3 : Edge uv is *strongly visible* from edge xy if and only if $yu"v"x$ are monotonic on edge xy

Proof : [**only if**] (refer to Figure 10(a)) Suppose edge uv is strongly visible from a point z on edge xy. ICH(y,u) must lie to the left of edge zu, and ICH(v,x) must lie to the right of edge zv. Since both ICH(y,u) and ICH(v,x) are concave in H, edge $uu"$ lies to the left of edge zu and edge $vv"$ lies to the right of edge zv. Therefore, $u"$ lies between y and z and $v"$ lies between z and x, thus $yu"v"x$ are monotonic on edge xy.

[**if**] (refer to Figure 10(b)) Suppose $yu"v"x$ are monotonic on edge xy. ICH(y,u) lies to the left of edge $u"u$ and ICH(v,x) lies to the right of edge $v"v$. Since edge $u"u$ does not intersect edge $v"v$, the Q($uvv"u"$) lies inside H, and inside P. Either one of the two diagonals of the Q lie inside. If edge $u"v$ lies inside Q($uvv"u"$), edge uv is completely visible from $u"$. If edge $v"u$ lies inside Q($uvv"u"$), edge uv is completely visible from $v"$. If both edges lie inside, edge uv is completely visible from any point w on edge $u"v"$. Therefore, edge uv is strongly visible from edge xy. ■



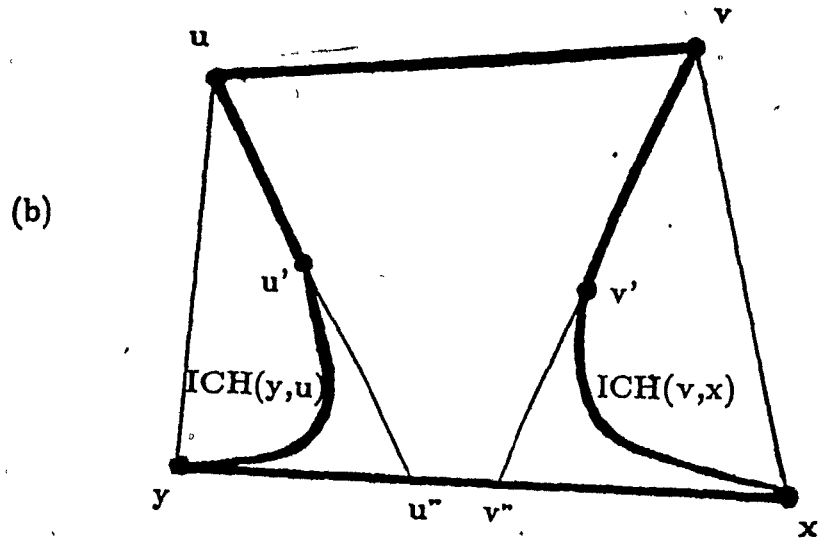


FIGURE 10

Lemma 5.4 : Edge uv is *completely visible* from edge xy if and only if $x=v"$ and $y=u"$.

Proof : [only if] Suppose edge uv is completely visible from edge xy. Then edge yu and edge vx must lie inside P. Thus $C(y,u)$ does not intersect edge yu, and $C(v,x)$ does not intersect edge vx. Therefore, $x = v"$ and $y = u"$

[if] Suppose $x = v"$ and $y = u"$. Then edge vx and edge yu lie inside P. Therefore an edge wz for any w on edge uv and any z on edge xy must also lie inside P. Therefore, edge uv is completely visible from edge xy. ■

Theorem 1 : Given a simple polygon P, and two edges uv and xy totally facing each other, ALGORITHM EDGE-TO-EDGE VISIBILITY determines the visibility between these two edges in linear time.

Proof : [Correctness] Step 2 determine whether or not $C(y,u)$ or $C(v,x)$ has a *cut* through the $Q(uvxy)$. It can be done by two simple tests for intersection of each chain with edge yu and edge vx along the vertices on the chain. By Lemma 3, if there is a *cut*, the program terminate with 'No visibility'. Step 3, 4 and 5 find the reduced chain $RC(y,u)$ and $RC(v,x)$ by Lemma 4. Since $C(y,u)$ lies to the right of, or on edge yu, $C(y,u)$ and edge yu can be considered as a simple polygon. Similarly, $C(v,x)$ and edge vx can also be considered as a simple polygon. Step 6 and 7 find the polygon H which can be done by any existing correct convex hull algorithm for a simple polygon [Pr&Sm]. Step 8 and 9 determine one of the four kind of visibility from edge uv to edge xy and from edge xy to edge uv by Lemma 5.1 to 5.4. The strongly and weakly visible region is between u" and v" (see Figure 9 (b), (c) and 10 (a), (b)). In the case of partial visibility, it is obvious that the visible region is between the two critical lines of support.

[Complexity] Since step 1, 2 and 3 are just linear scans along the chains, each can be done in linear time. In Step 4 and 5, each point being searched is deleted, and every intersection point is considered only once. Thus, this step takes linear

time. The convex hull of a simple polygon in step 6 can be found in linear time. Constructing the polygon H in step 7 takes only a constant time. Step 8 and 9 are just some linear time tests on the polygon H. The critical lines of support can be found either in $O(n)$ time with the 'rotating calipers' of Toussaint [To5] or in $O(\log n)$ time with the more recent algorithm of Rohnert [Rh]. Therefore, ALGORITHM EDGE-TO-EDGE VISIBILITY runs in linear time. ■

The computation of the reduced chain in step 4 and 5 can be done by a variety of alternate methods. However, most of them either use some very complicated data structures or require more than linear time. They will be discussed in the last section of this chapter.

3.32 Improper Edge-to-Edge Visibility

In this section, we consider the *improper edge-to-edge visibility* problem. Only the identification of the visibility region in the previous algorithm needs to be modified.

Lemma 6 : Given edge uv is partially facing edge xy and edge xy is totally facing edge uv

1) edge uv is partially visible from edge xy if and only if H is a simple polygon.
2) edge uv is weakly visible from edge xy if and only if $u"$ and $v"$ are on edge xy, edge $v'v"$ does not intersect ICH(y,u) and edge $u'u"$ does not intersect ICH(v,x).
3) edge uv is strongly visible from edge xy if and only if $yu"v"x$ are monotonic on line xy.
4) edge uv cannot be completely visible from edge xy
5) part 1 to part 4 of the lemma still hold if edge uv is totally facing edge xy and edge xy is partially facing edge uv.

Proof :

ALGORITHM IMPROPER EDGE-TO-EDGE VISIBILITY

Input : A simple polygon P=(p_1, p_2, ..., p_n), and two edges in P, uv and xy, where edge uv partially faces edge xy and edge xy totally faces edge uv.

Output : the visibility region between edge uv and edge xy.

Begin

1-7. same as ALGORITHM EDGE-TO-EDGE VISIBILITY

8. [*identify the visibility region of edge uv from edge xy*]
   If $yu''v''x$ are monotonic on line xy,
   then edge uv is *strongly visible* from edge xy between u'' and v'';
   If u'' and v'' is on edge xy and edge $v'v''$ does not intersect ICH(y,u) and edge $u'u''$ does not intersect ICH(v,x),
   then edge uv is *weakly visible* from edge xy between u'' and v'';
   If H is a simple polygon,
   then edge uv is *partially visible* from edge xy between L_{uv} and R_{uv} on edge uv and L_{xy} and R_{xy} on edge xy, where L_{uv}, R_{uv}, L_{xy}, and R_{xy} are defined in the same way as in step 8 of ALGORITHM EDGE-TO-EDGE VISIBILITY.

9. [*identify the visibility region of edge xy from edge uv*] Analogous to step 8 with x replacing u and v replacing y.

End

1-3) Parts 1, 2, and 3 are taken care of by Lemmas 5.1 to 5.3.

4) Since edge uv is partially facing edge xy, either x or y is not in RHP(u,v). So either x is not visible from u or y is not visible from v. Therefore, they cannot be completely visible from each other.

5) The proofs to part 1 and part 4 are exactly the same. Part 2 and Part 3 follow straightforwardly, with a note that ICH(v,x), a degenerate case, must be ~~edge~~ vx in order to satisfy the assumptions. With uvx concave in Q($uvxy$), only vertex v on edge uv can possibly see x. If edge xy is weakly or strongly visible from edge uv then x must be visible from v. Also if x'' is on edge uv or $vx''y''u$ are monotonic on line uv then x'' is v.

Theorem 2 : Given a simple polygon P and two edges, uv and xy, where edge uv

is partially facing edge xy and edge xy is totally facing edge uv, ALGORITHM IMPROPER EDGE-TO-EDGE VISIBILITY determines the visibility between these two edges in linear time.

Proof : Same as Theorem 1 except step 8 and 9 are referenced by Lemma 6 instead of Lemma 5.1 to 5.4.

3.33 Vertex-to-Edge Visibility

Thirdly, the vertex-to-edge visibility problem can be considered as a special case of an edge-to-edge visibility problem. Take the vertex w as an edge uv where u and v are w. We construct the polygon H with two chains and one edge.

ALGORITHM VERTEX-TO-EDGE VISIBILITY

Input : A simple polygon $P=(p_1, p_2, ..., p_n)$, two edges pq and wz, where only vertex w to edge pq visibility between edge pq and wz is possible.

Output : the visibility region between edge pq and edge wz.

Begin

0. [*transform edge pq and vertex w to edge xy and edge uv*] Let x be p, y be q, edge xy is edge pq, and let u be w, v be w. Edge uv has length 0.

1-6. same as ALGORITHM EDGE-TO-EDGE VISIBILITY

7. [*construct polygon* H] H ← ICH (wp), edge pq, ICH (qw)

8. [*identify the visibility region*]

   If $w = x''$ and $w = y''$,

   then edge pq is *strongly visible* from edge wz at w.

   If H is a simple polygon,

   then edge pq are *partially visible* from edge wz between u'' and v'' on edge pq and w.

End

Lemma 7 : Given two edges pq and wz of a simple polygon, where only vertex w to edge pq visibility between edge pq and wz is possible.

a) H is a simple polygon if and only if edge pq is partially visible from edge wz.

b) edge pq is weakly visible from edge wz if and only if edge pq is strongly visible from edge wz.

c) edge wz cannot be weakly visible from edge pq.

d) $w = p''$ and $w = q''$ if and only if edge pq is strongly visible from edge wz

e) edge wz cannot be strongly visible from edge pq

f) complete visibility is impossible.

Proof :

a) The proof is similar to that of lemma 5.1.

b) [only if] Suppose edge pq is weakly visible from edge wz. Since edge wz is not visible from edge pq except vertex w, edge pq must be strongly visible from w, or from edge wz.

[if] Weak visibility is implied by strong visibility.

c) Vertex z on edge wz is not visib e from edge pq.

d) [if] Suppose edge pq is strongly visible from edge wz. Since only w is visible from edge pq, edge pq is strongly visible from vertex w. Since ICH(p,w) and ICH(w,q) are concave in H, they cannot intersect triangle wpq except at the endpoints; otherwise, w is not visible from p and q. Therefore, ICH(p,w) is edge pw and ICH(w,q) is edge wq, and hence, $w = p'' = q''$.

[only if] Suppose $w = q''$ and $w = p''$. Then ICH(p,w) is edge pw and ICH(w,q) is edge wq. So w is visible from p and q. Therefore, edge pq is strongly visible from vertex w, or from edge wz.

e) It is implied by part c; not weakly visible implies not strongly visible.

f) It is implied by part e; not strongly visible implies not completely visible which is symmetrical. ■

Theorem 3 : Given a simple polygon P, and two edges pq and wz of a simple polygon, where only vertex w to edge pq visibility between edge pq and wz is possible. ALGORITHM VERTEX-TO-EDGE VISIBILITY determines the visibility between these two edges in linear time.

Proof : Lemma 3 and lemma 4 apply also to the vertex-to-edge case when we consider the vertex as an edge of same endpoints. Step 8 is also justified by Lemma 7. The rest of the proof is the same as that of Theorem 1.

3.34 Vertex-to-Vertex Visibility

Finally, in the *vertex-to-vertex visibility*, let r and s be the two vertices. It suffices to perform a linear scan for the intersection between edge rs and each edge of the polygon P, and to test whether edge rs is inside the polygon.

3.4 Alternative Methods

The reduced chain computation described in last section can be viewed as the problem of cutting a polygon with a straight line through two given vertices p and q of the polygon. It can be solved by many other ways.

One conceptually simple way is to find the intersection points between the polygon edges and edge pq and sort them along edge pq, which can be done in $O(n \log n)$ time [Kn]. Then a linear trace will report the reduced chain. This procedure is dominated by a sorting algorithm and thus requires $O(n \log n)$ running time.

A Jordan sequence is a sequence of real numbers $x_1, x_2, ..., x_n$ which are the abscissas of the intersection points of a Jordan curve in the plane with the x-axis, listed in the order the points occur in the Jordan curve. With some transformation in the plane, the reduced chain computation is equivalent to sorting the Jordan sequence. Sorting Jordan sequences has applications on intersecting a simple polygon with a line [Ho&Mh], [Ed] and computational geography [Ho&Mh&Rs&Ta]. Recently, the Jordan sequence sorting problem was solved in linear time in [Ho&Mh&Rs&Ta] and [Ho&Mh], by converting it into a data manipulation problem that involves repeated splitting of lists. This algorithm requires complicated and heavy data structures such as circularly level-linked 2-4 trees, an

extension of level-linked 2-4 trees invented by Huddleston & Mehlhorn [Hu&Mh] [Mh] and Mater & Salveter [Ma&Sl], which in turn is an extension of level-linked 2-3 tree [Br&Ta].

A similar procedure to that proposed in the last section using stacks, to do the reduced chain computation, was discovered independently in the context of proving the polygon-cutting theorem in [Ca]. The theorem states that every simple polygon can be decomposed into two simple polygons, each of them having at most two thirds of the total weight, defined as the sum of each assigned weight on the vertices.

The reduced chain computation described in the last section deletes *every* vertex *immediately* after the scanning in either direction on the chain. It is believed that this procedure is the simplest, most efficient and direct one.

CHAPTER FOUR
Conclusion and Further Research

We have given four natural definitions of visibility between a pair of edges in a simple polygon P. The edge-to-edge visibility can be determined indirectly by either the *Edge Visibility Polygon* of P or the shortest paths in P. The former finds only the weak and partial edge-to-edge visibility, while the latter finds only the partial edge-to-edge visibility. Both indirect approaches either run in $O(n \log n)$ time, or in $O(n)$ time but use the complicated linear polygon triangulation algorithm. We have presented a direct, efficient, simple, linear running time and therefore optimal algorithm to compute the four edge-to-edge visibilities: complete, strong, weak, and partial, and to report the visible region on both edges.

With some minor modifications, ALGORITHM VERTEX-TO-EDGE VISIBILITY in the last chapter can be used to determine the visibility between a vertex and an edge of a simple polygon. An edge is either completely or partially visible from a vertex in a simple polygon.

There are several interesting and closely related open problems in this area. Avis & Toussaint [Av&To] give a linear time optimal algorithm for determining whether or not a given polygon is completely, strongly or weakly visible from a given edge. A more difficult and interesting version of the problem is: given a polygon, does there exist an edge from which the polygon is completely, strongly or weakly visible? The straightforward approach is to apply the above algorithm in [Av&To] to each edge in turn, yielding an $O(n^2)$ algorithm. However, the complete and strong visibility case can be solved in linear time by using the kernel finding algorithm of Lee & Preparata [Le&Pr1]. It is an open question whether we can do better for the weak visibility case.

In the case where the polygon is not visible from any edge, it is natural to define an edge visibility polygon of a simple polygon from a given edge under the four edge-to-edge visibility definitions.

The weak edge visibility polygon discussed in chapter two can be found in $O(n \log n)$ time efficiently and practically [El2] [Le&Li] [Ca&Gu] and impractically in linear time theoretically [To1] with the linear polygon triangulation algorithm [Ta&Va]. A simpler linear time algorithm is desired for finding the weak edge visibility polygon.

The strong edge visibility polygon has no unique definition. Any subpolygon of the weak edge visibility polygon, containing one point on the given edge and having every edge strongly visible from that point, can be a strong edge visibility polygon. However, we can define the maximum area strong edge visibility polygon and the maximum perimeter strong edge visibility polygon. The former is the largest area that a stationary guard on the edge can see, and the latter is the longest perimeter polygon. Computations of these polygons also need further investigation.

The complete edge visibility polygon can easily be proved to be the intersection of the two vertex visibility polygons from the two endpoints. Any point lying outside the intersection cannot see either one of the endpoints. Any point lying inside the intersection can see any point of the edge since it can see both endpoints. Intersection of two subpolygons of a simple polygon can be done in linear time by scanning both lists of edges. Vertex visibility polygons can be found in $O(n)$ time [El&Av], so this problem can be solved in $O(n)$ time.

Consider the problem of computing the complete, strong, weak and partial edge visibility graphs of a simple polygon. Our linear time edge-to-edge visibility algorithm can identify *all four* graphs, but it is of not much help here since the application of the algorithm to all pairs of edges yields an $O(n^3)$ algorithm. On

the other hand, the weak edge visibility polygon can be used here for computing the partial and weak edge visibility graphs. For each edge of the polygon, we must compute the weak edge visibility polygon of which a linear scan now identifies all edges of the polygon partially or weakly visible from the edge. This results in an overall time complexity of $O(n^2 \log n)$ if the weak edge visibility polygon is obtained from [El2], [Le&Li] or [Ca&Gu] and of $O(n^2)$ if it is obtained from [Toi]. For the strong edge visibility graph, we have to use the brute force $O(n^3)$ algorithm above. The complete edge visibility graph can be computed in $O(n^2)$ time in the same way as computing the weak edge visibility graph but from the complete edge visibility polygon which can be found in $O(n)$ time. A similar $O(n^2)$ approach is to compute the vertex visibility polygon for every vertex of the polygon. Then we check the visibility between every pair of consecutive vertices to report an edge in the graph between the two edges in the polygon associated with the two pairs of consecutive vertices. This algorithm works properly since two edges are completely visible if and only if both endpoints of both edges are visible. Simpler, more efficient, and $O(n^2)$ algorithms to find all four graphs is another interesting research topic. $O(E)$ algorithms to generate the four visibility graphs, where E is the number of edges in the graph, would also be a great improvement.

Another interesting open question would be to determine whether every edge on the boundary of the polygon is completely, strongly, weakly, or partially visible from every, some or one edge of a given set of edges in the polygon. The corresponding problem for vertex visibility, that is, checking the visibility of a simple polygon from a set of polygons is discussed in [El1]. A generalized version of the visibility problem would be to determine a *minimal* set of edges from which every edge of the polygon is completely, strongly, weakly or partially visible from every edge in the set. It is known that in the worst case a guard may

have to visit $\lfloor \frac{n}{3} \rfloor$ locations in order to observe a n-sided polygon [Cv].

Finally, we note that all the problems in three dimensions corresponding to those discussed in this thesis are open.

Bibliography

[As] T. Asano, "Efficient algorithm for finding the visibility polygon for a polygonal region with holes", Tech. Rept., Osaka Electro Communication University, 1985.

[Av&El] D. Avis and H. ElGindy, "A combinatorical approach to polygon similarity", *IEEE Trans. Information Theory,* Vol. IT-29, 1983, pp. 148-150.

[Av&To] D. Avis and G.T. Toussaint, "An optimal algorithm for determining the visibility of a polygon from an edge", *IEEE Trans. Computers,* Vol. C-30, December 1981, pp. 910-914.

[Br&Ta] M.R. Brown and R.E. Tarjan, "Design and analysis of a data structure for representing sorted lists", *SIAM J. of Comput. 9,* 1980, pp. 594-614.

[Ca] B. Chazelle "A theorem on polygon cutting with applications", *23rd Annual IEEE Symposium on Foundations of Computer Science,* 1982, pp. 339-349

[Cu&Le] W.H. Chu and D.T. Lee, "The hidden line elimination problem revisited", *Proceedings of the International Conference on Advanced Automation,* Taipei, Taiwan, December 1983, pp. 469-480.

[Cv] V. Chvátal, "A Combinatorial theorem in plane geometry", *J. Combinatorial, Th B.,* Vol. 18, 1975, pp. 39-41.

[Ca&Gu] B. Chazelle and L.J. Guibas, "Visibility and intersection problems in plane geometry", *Proc. Symposium on Computational Geometry,* Baltimore, June 1985, pp. 135-146.

[Do] B.R. Donald, "Hypothesizing channels through free-space in solving the findpath problem", Massachussetts Institute of Technology, Artificial Intelligence Laboratory, A.I.Memo No. 738, June 1983.

[Da&Be] L. Davis and M. Benedikt, "Computational models of space: Isovists and isovist fields", *Comp. Graphics and Image Proc.,* Vol. 11, 1979, pp. 49-72.

[Ed] H. Edelsbrunner, Problem P36, *Bull EATCS 21,* October 1983, pp. 195.

[El1] H. ElGindy, "Visibility in polygons with applications", M. Sc. Thesis, School of Computer Science, McGill University, November 1980.

[El2] H. ElGindy, "Hierarchical decomposition of polygons with applications", Ph. D. Thesis, School of Computer Science, McGill University, May 1985.

[El&Av] H. ElGindy and D. Avis, "A linear algorithm for computing the visibility polygon from a point", *Journal of Algorithms,* Vol. 2, 1981, pp. 180-197.

[El&Av&To] H. ElGindy, D. Avis, and G.T. Toussaint, "Applications of a two-dimensional hidden-line algorithm to other geometric problems", *Computing*, Vol. 31, 1983, pp. 191-202.

[Fr&Lo] H. Freeman and P.P. Loutrel, "An algorithm for the two dimensional 'hidden line' problem", *IEEE Trans. Electronic Computers*, Vol. EC-16, No. 6, 1967, pp. 784-790.

[Gr] J.G. Griffiths, "Bibliography of hidden-line and hidden-surface algorithms", *Computer Aided Design 10*, No. 3, 1978, pp. 203-206.

[Ga&Jo&Pr&Ta] M.R. Garey, D.S. Johnson, F.P. Preparata and R.E. Tarjan, "Triangulating a simple polygon", *Information Processing Letters*, Vol. 7, 1978, pp. 175-179.

[Gü&Ra&Sl] L. Guibas, L. Ramshaw, and J. Stolfi, "A kinetic framework for computational geometry", *Proc. 24th Annual FOCS Symp* , 1983, pp. 100-111.

[Gh&Sy] S.K. Ghosh and R.K. Shyamasundar, "A linear time algorithm for computing the visibility polygon from an edge," *Proc of Seventh International Conference on Pattern Recognition*, 1984, pp. 471-474.

[Hi] D.W. Hightower, "A solution to line-routing problems on the continuous plane", *Proc. of Design Automation Workshop*, 1969, pp. 1-24.

[Ho&Mh] K. Hoffmann and K. Mehlhorn, "Intersecting a line and a simple polygon", *Bull EATCS 22*, February 1984, pp. 120-121.

[Ho&Mh&Rs&Ta] K. Hoffmann, K. Mehlhorn, P. Rosenstiehl and R.E. Tarjan "Sorting Jordan sequences in linear time", *Proc Symposium on Computational Geometry*, Baltimore, June 1985, pp. 196-203.

[Hu&Mh] S. Huddleston and K. Mehlhorn, "A new data structure for representing sorted lists", *Acta Info 17*, 1982, pp. 157-184.

[Kn] D.E. Knuth, *The Art of Computer Programming Volume III : Sorting and Searching*, Addison-Wesley, Reading, Mass., 1973.

[Le] D.T. Lee, "Visibility of a simple polygon", *Computer Vision, Graphics and Image Processing*, Vol. 22, 1983, pp. 207-221.

[Le&Li] D.T. Lee and A. Lin, "Computing the visibility polygon from an edge", Tech. Rept., Northwestern University, 1984.

[La&Ma] C. Lantuejoul and F. Maisonneuve, "Geodesic methods in quantitative image analysis", *Pattern Recognition*, Vol. 17, 1984, pp. 177-187.

[Le&Pr1] D.T. Lee and F.P. Preparata, "An optimal algorithm for finding the kernel of a polygon", *JACM*, Vol. 26, No. 3, 1979, pp. 415-421.

[Le&Pr2] D.T. Lee and F.P. Preparata, "Euclidean shortest paths in the presence of rectilinear barriers", *Networks*, Vol. 14, 1984, pp. 393-410.

[Le&Pr3] D.T. Lee and F.P. Preparata, "Location of a point in a planar subdivision and its applications", *SIAM J. Computing,* Vol. 6, 1977, pp. 594-606.

[Lo&We] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles", *Communications of the ACM,* Vol. 22, 1979, pp. 560-570.

[Ma&Sl] D. Mater and C. Salveter, "Hysterical B-trees", *Info. Proc. Lett. 12,* 1981, pp. 199-202.

[Mh] K. Mehlhorn, *Data Structures and Efficient Algorithms, Vol. 1, Sorting and Searching,* Springer Verlag, 1984.

[Ml] A.A. Melkman, "On computing the visibility polygon from a point", Tech. Rept. CS-85-260, Ben Gurion University, Israel, April 1985.

[Nl] N.J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques", *Proceedings, IJCAI-69,* pp. 509-520.

[Pr&Sm] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction,* Springer-Verlag, 1985, pp. 160-165.

[Ra] D. Rappaport, "A linear algorithm for eliminating hidden line from a polygonal cylinder", *The Visual Computer,* Vol. 2, No. 2, 1986.

[Ra&To] D. Rappaport and G.T. Toussaint, "A simple linear hidden-line algorithm for star-shaped polygons", *Pattern Recognition Letters,* Vol. 3, 1985, pp. 35-39.

[Rh] H. Rohnert, "Shortest paths in the plane with convex polygonal obstacles", Tech. Rept. A 85/06, University of Saarbrücken, 1985.

[Sm] M.I. Shamos, *Problems in computational geometry,* Carnegie Mellon University, (1975) revised 1977.

[Sp&Ha1] L.G. Shapiro and R.M. Haralick, "Algorithms for inexact matching", *Proc. 5th Int Conf. on Pattern Recognition,* Miami Beach, 1980, pp. 202-207.

[Sp&Ha2] L.G. Shapiro and R.M. Haralick, "Decomposition of two dimensional shapes by graph-theoretic clustering", *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-1, January 1979, pp. 10-20.

[Su&OR] S. Suri, J. O'Rourke, "Worst-case optimal algorithms for constructing visibility polygons with holes", Tech. Rept. JHU/EECS-1985/12, August 1985.

[Sc&To] J.-R. Sack and G.T. Toussaint, "Translating polygons in the plane", Tech. Rept. SCS-TR-47, Carleton University, March 1984.

[Se&Wl] J. Stoer and C. Witzgall, *Convexity and Optimization in Finite Dimensions I,* Springer-Verlag, 1970.

[To1] G.T. Toussaint, "A linear time algorithm for solving the strong hidden-line problem in a simple polygon", Tech. Rept. SOCS-86.2, McGill University, January 1986.

[To2] G.T. Toussaint, "Shortest path solves edge-to-edge visibility in a polygon", Tech. Rept. SOCS-85.19, McGill University, September 1985, also to appear in *Pattern Recognition Letters.*

[To3] G.T. Toussaint, "Pattern recognition and geometrical complexity", *Proc. 5th Int. Conf. on Pattern Recognition,* Miami Beach, FL., December 1980.

[To4] G.T. Toussaint, "Shortest path solves translation separability of polygons", Tech. Rept. SOCS-85.27., McGill University, October 1985.

[To5] G.T. Toussaint, "Solving geometric problems with the rotating callpers", *Proc. MELECON '83,* Athens, Greece, 1983.

[To&El] G.T. Toussaint and H. ElGindy, "Separation of two monotone polygons in linear time", *Robotica,* Vol. 2, 1984, pp. 215-220.

[To&Sc] G.T. Toussaint and J.-R. Sack, "Some new results on moving polygons in the plane", *Proc. Robotic Intellignece and Productivity Conference,* Detroit, 1983, pp. 158-163.

[Ta&Va] R.E. Tarjan and C. Van Wyk, "A linear time algorithm for triangulating simple polygons", manuscript, October 1985.

[Ya] F.F Yao, "On the priority approach to hidden surface algorithms", *Proc. 21st Annual Symposium on Foundations of Computer Science,* October 1983, pp. 301-307.