APPLICATIONS OF VISIBILITY SPACE IN POLYGON SEARCH PROBLEMS

by

Zhong Zhang

M.Sc., Simon Fraser University University, 1998

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

APPROVAL

Name:	Zhong Zhang
Degree:	Doctor of Philosophy
Title of thesis:	Applications of Visibility Space in Polygon Search Problems

Examining Committee: Dr. Valentine Kabanets
Chair

Dr. Tiko Kameda
Senior Supervisor

Dr. Qianping Gu
Senior Supervisor

Dr. Binay Bhattacharya
Supervisor

Dr. Ramesh Krishnamurti
SFU Examiner

Dr. Masafumi Yamashita
Professor of Computer Science
Kyushu University

Date Approved: July 14, 2005

Abstract

This thesis investigates several problems related to searching a polygonal area for intruders.

The mutual visibility between an arbitrary pair of points on the boundary of a polygon is an important piece of information we can make use of when searching a polygon. We extensively employ the visibility diagram that represents mutual visibility information for each pair of boundary points.

We first investigate the *two-guard room search* problem, where two guards cooperate in finding an intruder by maintaining mutual visibility. In terms of the visibility diagram we characterize the class of searchable rooms in a concise way. We also find all doors in a polygon, if any, such that the resultant rooms are searchable by two guards. The second problem we tackle in this thesis is the polygon search problem by a *boundary 1-searcher*, who moves along the boundary of a polygon and can see the points on the beam from a flashlight. We identify the patterns that make a polygon non-searchable. The third problem we investigate is to search a polygon with one hole by two boundary 1-searchers. We solve this problem by extending the visibility diagram.

Acknowledgments

I wish first to express my sincere gratitude to Dr. Tiko Kameda, for the guidance, encouragement, support and family-like environment that he has given me over the past years. Thank you so much, Tiko, for leading me into the field, touching me with your intellectual passion, and mentoring me with your academic seriousness. Thank you for telling me that I can do it, always.

I would like to thank Dr. Qianping Gu for his unique advice and sincere support during my studies toward this thesis. He has given me a great amount of advice as how to select a good topic, how to collect resources for it, and how to approach it.

Special thanks also go to Dr. Binay Bhattacharya. He has taught me many techniques, providing fresh insight and critical comments on my approach to the problems I tried to solve in my thesis.

I should thank Dr. Ramesh Krishnamurti for spending a great amount of time reviewing my thesis as an internal examiner. I really appreciate the comments he has made. This greatly helped me when I was finalizing my thesis.

Dr. Masafumi Yamashita has provided with my valuable comments during his visits to our lab. His sharpness, insightful expertise, and visionary viewpoints have polished my knowledge and enlarged my scope of the topic. His role as an external examiner has provided a unique chance for me to reexamine my thesis in a more significant way.

My thanks should be given to the supporting staff in the School of Computing Science at Simon Fraser University. I should also thank the fellow classmates in the School for talking and discussing with me many interesting problems during my research.

Finally, I am deeply grateful to my family. To Fang and Zhirui, who have suffered from the separation with me, to my Mom and Dad, and to my Mom- and Dad-in-law, for your patience, support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Background

As a topological entity a *simple polygon* is merely a closed non-intersecting loop of line segments. However, if we treat it as a metric entity, its boundary structure can be extremely complex. Various computational problems are the result of efforts to classify polygons with respect to their boundary structures.

Polygon search problems are one of the different incarnations of the various *visibility* problems, which are fundamental in computational geometry. The solutions to them can help us better understand the inherent nature of the visibility in simple polygons. Because of this, they recently attract more and more attention in the research community in computational geometry. Furthermore, since these problems exist not only in computer science, but also in other areas, such as robotics and mathematics, they are also interesting to the researchers from these fields.

1.1 Introduction

In a nutshell, the *polygon search problem* is the problem of searching for mobile intruders inside a polygonal region by one or more mobile searchers or guards. The problem was first proposed and discussed by Suzuki and Yamashita [106]. Before describing the notation to be used in our discussions and reviewing the previous work on the problem, we first briefly introduce the *Art Gallery Problem.* It should be noted that for many terms and expressions, we just use them first on the assumption that many of them are common sense. We shall define them formally later on when our discussions unfold.

During a conference at Stanford in August 1976, in response to the request from V.

Chvátal who asked for an interesting geometric problem, V. Klee posed the following question: How many guards are necessary and sufficient to guard the artistic objects in a given art gallery with n walls? The name *Art Gallery Problem* was coined for this problem. Chvátal soon [31] established that $\lfloor n/3 \rfloor$ guards are *occasionally* necessary and *always* sufficient to guard an art gallery represented by a simple polygon with n vertices. For a detailed proof of the theorem, see [81]. As an example, we show a polygon with 10 vertices that is guarded by $\lfloor 10/3 \rfloor = 3$ guards in Figure 1.1 (Their positions are represented by the points marked as g.).

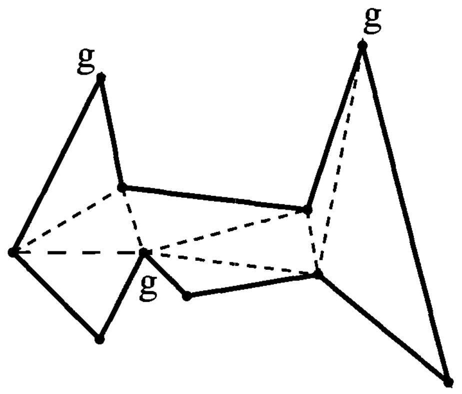


Figure 1.1: Guarding a polygon.

Since the establishment of Chvátal's theorem, a tremendous amount of research on the Art Gallery Problem has been carried out. This superficially naïve but inherently complex problem has been not only extended by mathematicians in several directions, but also further studied by computer scientists. Many variations have been proposed and studied and a great number of research results have been published. In 1987, O'Rourke published a book entitled *Art Gallery Theorems and Algorithms* [81], the first documentation dedicated solely to the topic. The book further inspired interests in the topic. In 1992, Shermer [98] conducted a thorough survey in *Recent Results in Art Galleries.* In 2000, Urrutia published another survey [113], which can be considered as complementary to O'Rourke's book and Shermer's paper. Besides these surveys, numerous research papers were published on the topic during the past decades. Due to the huge number of the papers, we cannot list them here. But the aforementioned book [81] and two papers [98, 113] give us excellent surveys on the subject. Interested readers are referred to them.

Essentially speaking, the Art Gallery Problem and its variations deal with polygon guarding problems with guards that are stationary at their prespecified positions. Even for *diagonal* or *edge guards*, who can only move along a diagonal or an edge of a polygon, we can still regard them as stationary "fluorescent" lights, fastened on the diagonal or edge.

The polygon search problem is closely related to the Art Gallery Problem. Actually in many previous papers on the topic, authors have always mentioned the Art Gallery Problem in their introductions. In the polygon search problems, guards are capable of moving at a limited speed, i.e. the guards are *mobile.* In these problems, the guards are actually *searching* a polygon. So in many occasions, the guards are also called *searchers.*

Intuitively speaking, as searchers are able to move, the number of searchers required to search a polygon is drastically reduced. But in the meanwhile, the complexity of the problem is increased.

1.2 Preliminaries

We will use the 2-dimensional Euclidean plane as our coordinate system. In this system, a *point* is represented as an ordered pair of coordinates in different dimensions. Straight lines are the shortest distance between any two points. A *line* is of infinite length in both directions, while a *line segment* is a continuous portion of a line with two end points. If a line segment's two end points are u and v, we denote it as $\overline{uv}$. We use the Euclidean distance to measure the length of a line segment.

A polygon P is composed by an ordered sequence of points p_0, p_1, $\cdots$, p_{n-1}, $n \geq 3$, which are called the *vertices* of P, and n line segments from p_i to p_{i+1}, $i = 0, 1, \cdots, n-2$ and from p_{n-1} to p_0, which are called the *edges* of P. We assume that the order of the vertices is in the *clockwise direction* starting at p_0.

In this thesis, we assume that the polygons are *simple*, i.e., the only pairs of edges that intersect are pairs of consecutive edges, whose intersection must be exactly their common vertex.

For a simple polygon P, its edges form its *boundary*, which is denoted as ∂P. We assume that $\partial P \subseteq P$. We denote the length of the boundary as $|\partial P|$, which is the sum of the lengths of all the edges of P.

A *polygonal chain* (or just *chain*) is a continuous section of ∂P. The open (closed, resp.) *clockwise chain* on ∂P from u to v is denoted by $\partial P_{cw}(u, v)$ ($\partial P_{cw}[u, v]$, resp.), where

$u, v \in \partial P$. Half-open clockwise chains are denoted by $\partial P_{cw}[u, v)$ or $\partial P_{cw}(u, v]$, depending on which end is open. Similarly, the open (closed, resp.) *counter-clockwise chain* on ∂P from u' to v' is denoted by $\partial P_{ccw}(u', v')$ ($\partial P_{ccw}[u', v']$, resp.), where $u', v' \in \partial P$. Also, half-open counter-clockwise chains are denoted by $\partial P_{ccw}[u', v')$ or $\partial P_{ccw}(u', v']$, depending on which end is open. Note that the starting point or the ending point of a chain might not be a vertex of P.

Suppose that $v \in \partial P_{cw}(u, w)$. We write $u \prec_{cw} v$ and $v \prec_{cw} w$. If $u \prec_{cw} v$, we also say that $v \succ_{cw} u$. For a vertex v on ∂P, *Prec*(v) is the vertex immediately preceding v while *Succ*(v) denotes the vertex immediately succeeding v in the clockwise direction.

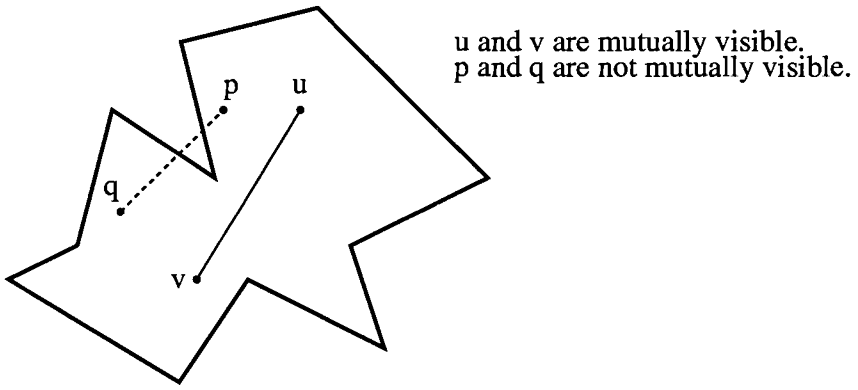

u and v are mutually visible.
p and q are not mutually visible.
p
u
q
v

Figure 1.2: Visibility in a polygon.

Two points $u, v \in P$ are said to be *mutually visible* if the line segment $\overline{uv}$ is completely contained inside P. Recall that $\partial P \subseteq P$. This definition allows the segment of visibility $\overline{uv}$ to go through a reflex vertex or graze along a polygon edge. Sometimes we also say that u *sees* v and vice versa, if u and v are mutually visible. For example, in Figure 1.2, u and v are mutually visible while p and q are not. Note that in Chapter 5, we use a slightly different definition of mutual visibility.

We say that a vertex of a polygon is *convex* if the interior angle between its two incident edges is at most 180^o; otherwise we say that it is *reflex*. Sometimes, if a vertex is convex, we also call it *non-reflex*.

We use Figure 1.3 as a reference for the following discussions. Let r be a reflex vertex. Its *backward ray* is the ray generated from *Succ*(r) to r while the *backward ray end point* is the point on ∂P at which the ray leaves the polygon for the first time and is denoted as

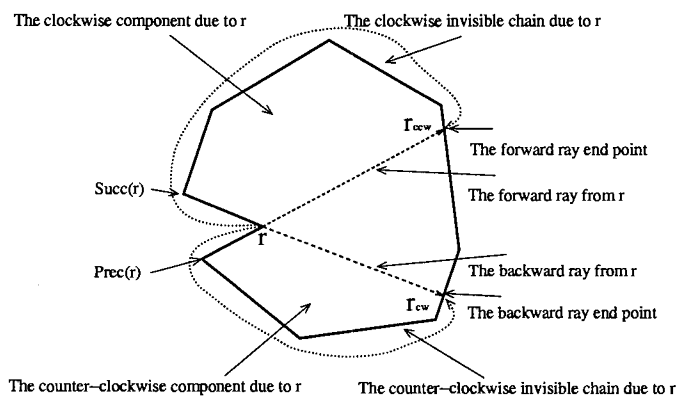

The clockwise component due to r
The clockwise invisible chain due to r
r_{ccw}
The forward ray end point
The forward ray from r
Succ(r)
r
Prec(r)
The backward ray from r
r_{cw}
The backward ray end point
The counter–clockwise component due to r
The counter–clockwise invisible chain due to r

Figure 1.3: Definitions related a reflex vertex r.

r_{cw}. We call the polygonal area formed by $\partial P_{cw}[r, r_{cw}]$ and the chord $\overline{rr_{cw}}$ the *clockwise component* w.r.t. r and denote it by $P_{cw}(r)$. Sometimes we also say that the clockwise component $P_{cw}(r)$ is due to r.

Similarly, the *forward ray* from r is the ray generated from *Prec*(r) to r while the *forward ray end point* of r is the point on ∂P at which the ray leaves the polygon for the first time and is denoted as r_{ccw}. We call the polygonal area formed by $\partial P_{cw}[r_{ccw}, r]$ and the chord $\overline{rr_{ccw}}$ the *counter-clockwise component* w.r.t. r and denote it by $P_{ccw}(r)$. Again, sometimes we also say that the counter-clockwise component $P_{ccw}(r)$ is due to r. A component can be completely contained in another component. A component is *non-redundant* if it does not contain any other components.

From the definitions of ray end points, we immediately know that for r_{cw} (r_{ccw}, resp.), the polygonal chain $\partial P_{ccw}(r, r_{cw})$ ($\partial P_{cw}(r, r_{ccw})$, resp.) is invisible to any point from r (excluding r) (*Prec(r)*, resp.) to *Succ(r)* (r (excluding r), resp.). We call it the *counter-clockwise (clockwise, resp.) invisible chain* due to r. It is easy to see that a reflex vertex r gives rise to two invisible chains and they do not have any common intersection.

Visibility graph is an important concept in computational geometry. Here we briefly discuss it. We first give some definitions in graph theory. A *graph* $G(V, E)$ consists of a set of elements V called the *vertices* of $G(V, E)$ and a set of pairs of vertices E called the *edges* of $G(V, E)$. Two vertices u and v in $G(V, E)$ are called *adjacent* if the pair (u, v) is an element in E (u and v are called the *endpoints* of the edge).

The *visibility graph* *VG*(P) of a given polygon P is the graph whose vertex set is the set of vertices of P. Two vertices u and v are adjacent in *VG*(P) if they are mutually visible in

P.

A *visibility polygon* is defined as follows. For a point $p \in P$, let $V(p)$ denote the set of all points in P that are visible from p. It is easy to see that the points in $V(p)$ form a polygon. We call $V(p)$ the visibility polygon of p. Note that p can be on ∂P.

A *polygon with hole(s)* is a polygon with the interior of one or more simple polygons removed from it. The definition of visibility between two points in a polygon with hole(s) is the same as the one in a polygon without holes.

We will use some other terms and expressions in our discussions in this thesis. But we will leave them until the relevant topic comes up for discussion.

1.3 Previous work

Many variations of the polygon search problem have been proposed and studied in the literature since its first proposal by Suzuki and Yamashita [106].

We first introduce the terminology for polygon search problems. We assume that there are *intruders* who move around in a polygon P. Without losing generality, we may assume that there is only one intruder. Our task is to use a set of searcher(s) to search the polygon such that the intruder is eventually detected or leaves the polygon, i.e., we are sure that the polygon is clear of the intruder. In the following discussions in this thesis, we interchangeably use *searchers*, *pursuers*, *hunters* and *guards* if no ambiguity arises.

Let $e(t) \in P$ denote the position of the intruder at time $t \geq 0$, and let $e : [0, \infty) \rightarrow P$ be a continuous function, representing his move path. The intruder is capable of moving arbitrarily faster than the searcher(s) (in order to make the problem non-trivial). The initial position $e(0)$ and the move path e are unknown to the searcher(s). Any region in P that might contain the intruder is referred to as *contaminated*; otherwise it is referred to as *clear*. If a region was contaminated, became cleared, and then becomes contaminated again, it is referred to as *recontaminated*.

Let N be the number of searchers. Denote by γ a *search schedule* or just a *schedule* of the searcher(s). It contains the specification of a continuous path for each searcher: $\gamma = \{\gamma^1, \gamma^2, \cdots, \gamma^N\}$. To be more specific, γ^i denotes the continuous path of the i^{th} searcher in the form $\gamma^i : [0, \infty) \rightarrow P$. Denote by $\gamma^i(t)$ the position of the i^{th} searcher at time $t \geq 0$.

A search schedule, γ, is called a *solution schedule* if for every continuous function $e : [0, \infty) \rightarrow P$ there exists a time $t \in [0, \infty)$ and an $i \in \{1, 2, \cdots, N\}$ such that $e(t) \in$

$VR(\gamma^i(t))$, where $VR(\gamma^i(t))$ is the visibility range of searcher i. We will define $VR(\gamma^i(t))$ later when we discuss different search problems. Sometimes, we may call a solution schedule using some other names, such as a *legal schedule*, a *winning schedule*, etc.

1.3.1 The watchman route problem

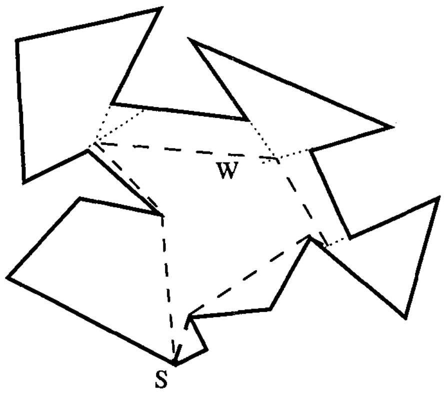


Figure 1.4: A watchman route.

The *Watchman Route Problem* was first introduced in [29] by Chin *et al.*. Suppose that a watchman has to patrol a polygon P. To do this, she has to find a closed "walk" W starting and ending at a *starting point* s such that every point in P is visible from some point in W. See Figure 1.4 for an example. In the figure, the dashed curve inside the polygon represents a watchman route. In order to minimize the cost, usually measured as the distance the watchman travels, it is desirable to find a shortest route. Note that the visibility range of the watchman in this problem is 360^o.

Ntafos *et al.* [80] proposed and studied the *External Route Problem.* Given a set of polygonal obstacles in the plane, the problem is to find a route along which a watchman follows to patrol the exterior of the obstacles such that each point in the exterior is visible from some point on the route.

As a restricted version of the External Route Problem, Gewali *at el.* [51] investigated the problem for computing the shortest watchman route for a *pair* of convex polygons.

Some other work regarding the Watchman Route Problem can be found in [25, 26]. Also the work in [23, 24] discussed how to find the shortest watchman route.

1.3.2 The Hunter's Problem

Suzuki and Yamashita [106] considered a polygon search problem in which both the searcher and the intruder are allowed to move freely within a given polygonal region. The searcher moves at a bounded speed while the intruder can move at an unbounded speed. Thus the searcher needs to figure out a search schedule in order to catch the intruder.

Many variations of this problem have been studied, such as that the searcher has a visibility range of 360^o, or the searcher has a fixed number of k searchlights, where the visibility range of the searcher is restricted to the k rays from the flashlights. The former is called the ∞-searcher problem while the latter is called the k-searcher problem. It was shown that there are some polygons that are searchable by a 2-searcher but not by a 1-searcher.

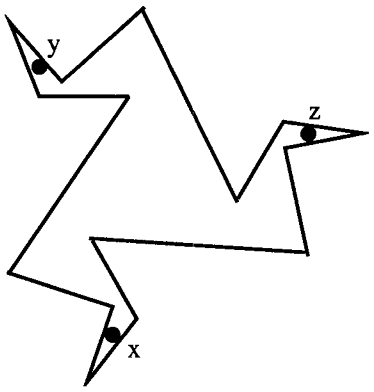


Figure 1.5: A polygon that is not searchable by a 1-searcher.

The work also established some necessary conditions for a polygon to be searchable. A polygon is called k-searchable (there is only one k-searcher) if, no matter what the given initial configuration of the intruder and the k-searcher is, it is always possible to move the searcher such that at some time the intruder will be falling into the visibility range of the searcher. For example, it was proven that if a polygon is 1-searchable, it has no three points such that the shortest path between any two of them is invisible to the third one. According to this, the polygon shown in Figure 1.5 is not 1-searchable, since point z cannot see any point on the shortest path between points x and y. The same situation applies to the other combinations where we consider the shortest paths between points x and z and points y and z, respectively.

In the sense that, when an intruder slips into a polygon, the searcher's task is to catch him, the above problem was named as *the Hunter's Problem* [98].

Urrutia [113] proved the following theorem regarding the number of the searchers.

Theorem 1.3.1 *$O(\ln n)$ searchers are always sufficient, and occasionally necessary to catch an intruder in any polygon with n vertices.*

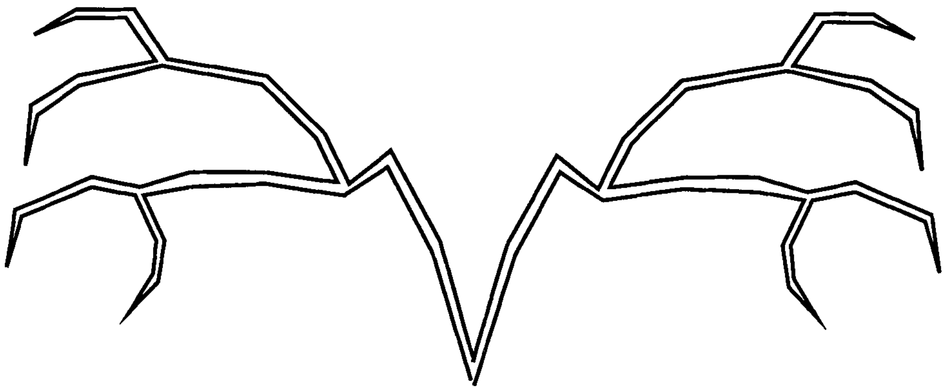


Figure 1.6: A binary polygon.

Figure 1.6 shows a family of polygons that actually require $O(\ln n)$ searchers. We call them *binary polygons* due to the similarity between their shape and a *binary tree.*

Since its first appearance, there have been a number of variations of the Hunter's Problem, though some of them do not use the problem name exactly or explicitly.

1.3.3 The two-guard problem

There are great interests in the two-guard street problem. One can imagine that there is a polygon with two prespecified ends, one called the *entrance* while the other called the *exit*, which divide the boundary of the polygon into two sides. Usually the entrance and the exit are the vertices of the polygon.

Two searchers (or guards), with one on each side of the street, start from the entrance, move along the two sides, and finally meet at the exit. It is required that during the search process, the two guards be always mutually visible. Thus in this problem, the visibility range of the guards is restricted to the line segment (represented by a flashlight beam) in between them. If there was an intruder slipping into the street, it is hoped that he would be eventually pushed to the exit and leave the street from there. In some literature, the two-guard problem is also called the *corridor problem.* Sometimes the entrance and the exit can be an edge rather than a vertex.

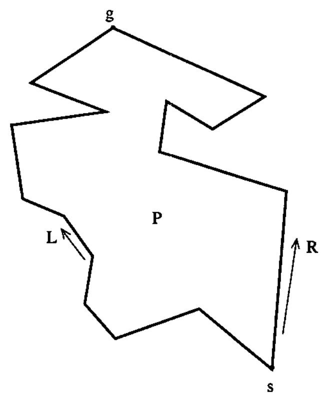


Figure 1.7: A polygon P with two sides L and R.

The two-guard street problem was first studied by Icking *et al.* [57]. Assume that we are given a simple polygon with n edges and two distinguished vertices s (start) and g (goal), representing the entrance and the exit, respectively; then the polygon's boundary consists of two polygonal chains, L and R, with common endpoints s and g. Both chains are oriented from s to g. See Figure 1.7 for an example. It is required that L and R be *mutually weakly visible.* L (R, resp.) is said to be weakly visible from R (L, resp.) if for each point $p \in L$ ($q \in R$, resp.) there exists a point $q \in R$ ($p \in L$) such that q and p are visible. Further definitions regarding the problem are given below.

A *walk* on P is a pair (l, r) of continuous functions such that: (1) $l : [0,1] \rightarrow L$, $r : [0,1] \rightarrow R$; (2) $l(0) = r(0) = s, l(1) = r(1) = g$; (3) $l(t)$ is visible from $r(t)$ for all $t \in [0,1]$. Any line segment $\overline{l(t)r(t)}$ is called a *walk line segment* of the walk. Note that a walk line segment must always lie within the street. The point $r(t)$ is the *walk partner* of $l(t)$, and vice versa. A walk on P is called *straight* if both l and r are non-decreasing with respect to the orientation of L and R. P is called *(straight) walkable* if it admits a (straight) walk. As a counterpart, a *counter-walk* on L and R is defined to be just like a walk, but with the boundary conditions $l(0) = s$, $l(1) = g$, $r(0) = g$, $r(1) = s$. A *straight* counter-walk has a non-decreasing function l and a non-increasing function r w.r.t.the orientation of L and R. P is called *(straight) counter-walkable* if it admits a (straight) counter-walk.

The *general walk* problem asks for a walk in which the total distance traveled by the two guards is minimized, i.e., the one where the sum of the lengths of the two curves given by the functions l and r becomes minimum.

A *walk instruction* [57], is defined to be one of the two elementary moves: (1) Both searchers move forward along segments of single edges; (2) One of the searchers moves forward, while the other moves backward along single edges. Clearly, any straight walk, if one exists for a given polygon, is of the minimum length.

The major contribution in [57] is stated in the following theorem.

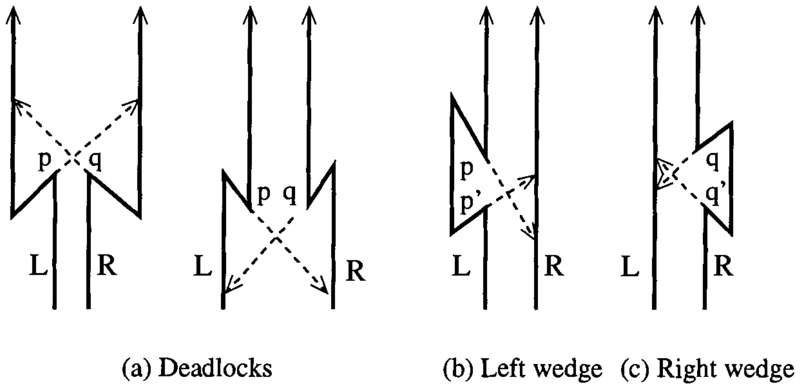


Figure 1.8: Three configurations for which no straight walks exist.

Theorem 1.3.2 *Let the chains L and R be mutually weakly visible. Then there is a straight walk for P if and only if none of the three configurations shown in Figure 1.8 exists for L and R. To test the conditions, and to construct a straight walk,* $O(n \log n)$ *time and linear space are sufficient.*

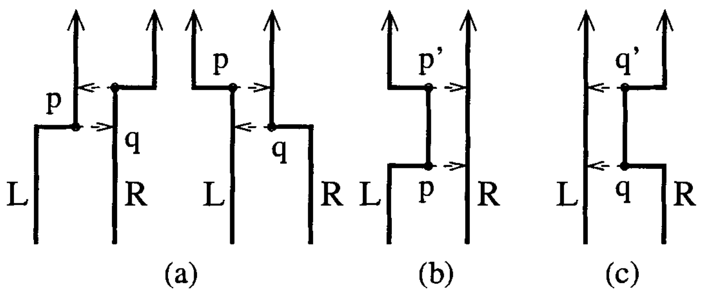


Figure 1.9: Three configurations for which no straight counter-walks exist.

The necessary and sufficient condition for a straight counter-walk in a street is stated in the following theorem [57].

Theorem 1.3.3 *There is a straight counter-walk if and only if the chains L and R are mutually weakly visible and none of the three configurations shown in Figure 1.9 exists. Such a straight counter-walk can be computed in $O(n \log n)$ time.*

For the general walk problem, the following result was obtained [57].

Theorem 1.3.4 *There is a walk for P if and only if the chains L and R are mutually weakly visible and P does not contain a deadlock, shown in Figure 1.8 (a). Furthermore, a walk of minimum length can be computed in time $O(n \log n + k)$ and in space $O(n)$, where k is the number of walk instructions.*

There have been some followups since then. Observing that some rays are *dominated* by others, Heffernan [56] proposed to build balanced trees using only *non-dominated* rays, while the trees' structure is similar to the one used in [57]. Furthermore, with the help of *shortest paths* and *shortest path trees*, the time complexity of Theorems 1.3.2 and 1.3.3 is improved to $\theta(n)$. However, it was asked whether there was an $O(k)$ algorithm for Theorem 1.3.4.

In the two-guard street problem, the starting point s and the ending point g are prespecified. Tseng *et al.* [110] considered the street problem from another perspective. They asked whether, for a given polygon with n vertices, there exists a pair of vertices such that the resultant street admits a general walk, a straight walk or a straight counter-walk. They obtained an $O(n \log n)$ time algorithm to find all such pairs of entrance and exit that admit a (straight) walk and an $O(n \log n) + m$ (m could be as large as $O(n^2)$.) time algorithm to find the pairs that admit a (straight) counter-walk.

Bhattacharya *et al.* [15] further reduced the above time complexity to $O(n)$ to find all the pairs of vertices such that the resultant streets are walkable. In their notation, given a street with the entrance s and the exit t, an *s-deadlock* (a *t-deadlock*, resp.) happens if the two searchers, both starting at s (t, resp.) and moving along the street boundaries in the opposite directions, have reached two reflex vertices such that neither of them can move forward without losing their mutual visibility. These two reflex vertices form an s-deadlock (a t-deadlock, resp.). The technique employed is to make use of a nice property of a class of LR-visible polygons [14, 36] (Also see below in Section 1.3.11.) to compute the s-deadlocks and t-deadlocks in linear time. The property is that in an LR-visible polygon, all the non-redundant components can be calculated in $O(n)$ time. Later on we also make use of this property in our work.

The work [14, 33, 58] also discussed the street search problem in other directions.

1.3.4 The room search problem

Suppose that an intruder slips into a *room* $P(d)$, which is a polygon P with a designated point d on its boundary, called a *door*. The door is a point where a searcher starts her search in order not to let the intruder escape through it. The goal of the searcher is to catch the intruder. This problem is called the *room search problem*. There are several papers discussing the problem under different search models.

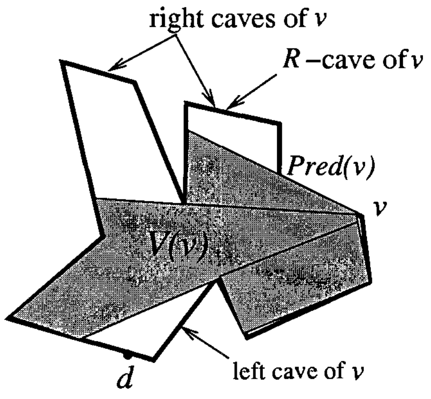


Figure 1.10: Definition of caves.

Lee *et al.* [74] discussed the room search problem using a 1-searcher. The work introduced a concept called *cave* under the observation that reflex vertices are the sources where the visibility difficulties arise. A *cave* is defined for a vertex v to be a maximal connected boundary chain $C(p,q)$ that is not a subset of $V(v)$, the visibility polygon of the vertex v. Note that two endpoints p and q of $C(p,q)$ lie on the directed line $\overrightarrow{vp}$ and $C(p,q)$ lies entirely to the left (called a *left cave*) or right (called a *right cave*) of $\overrightarrow{vp}$. A left cave $C(p,q)$ of v is called the *L-cave* of v if $p = Succ(v)$. Analogously, a right cave $C(p,q)$ of v is called the *R-cave* of v if $q = Prec(v)$. Figure 1.10 illustrates these definitions.

In order to analyze the problem, the following definitions are introduced. Let $C = \langle v_1, v_2, v_3 \rangle$ be a triple of vertices such that $v_1 \prec_{cw} v_2 \prec_{cw} v_3$. We call C an *s-triple* if v_1

has the L-cave and v_2 and v_3 lie in it, and v_3 has the R-cave and v_1 and v_2 lie in it. An s-triple C such that v_2 has the R-cave and v_1 and d (the door) lie in it is called an *l-triple.* Symmetrically, an s-triple C such that v_2 has the L-cave and v_3 and d lie in it is called an *r-triple.* For the sake of space, we do not go into the details of the paper.

The main theorem in [74] is as follows. Interested readers are referred to the paper.

Theorem 1.3.5 *A room $P(d)$ is 1-searchable if and only if the following three statements hold.*

(N1) There are no vertices a and b such that d and b lie in the L-cave or R-cave of a, and d and a lie in the L-cave or R-cave of b.

(N2) There is no s-triple $\langle a_1, a_2, a_3 \rangle$ such that a_1 and a_3 lie in the L-cave or R-cave of a_2.

(N3) There are no l-triple $\langle a_1, a_2, a_3 \rangle$ and r-triple $\langle b_1, b_2, b_3 \rangle$ such that $a_2 \prec b_2$ and every vertex v lying between a_2 and b_2 has an s-pair $\langle v_L, v_R \rangle$.

Park *et al.* [89] considered the same problem using two guards (with the same visibility requirement as the one for the two-guard street search problem, i.e., the two guards are always mutually visible). However, a search schedule for a room is different from the one for a street. When searching a street, the starting and ending points are prespecified, and at any time the two guards cannot go across the ending point. However, there is no such a condition in the room search problem. Even though the search schedule ends at some point g', during the search process the two guards may go across it. It was shown that there exist some polygons with one door which are not walkable as a street no matter where the ending point is, but can be searched by two guards as a room.

The street search problem and the room search problem were compared in [74]. If a street is searchable by two guards (while maintaining their mutual visibility), we can construct the search schedule. By following the same schedule, a 1-searcher can also search the street by moving the searcher along L and aiming its flashlight at the positions of the second guard on R. On the other hand, any 1-searchable street can also be searched by two guards, which was demonstrated by showing that any 1-searchable street also satisfies the conditions in Theorem 1.3.4. The catch is that a 1-searcher still cannot clear a deadlock. As for the room search problem, if a room is searchable by two guards (while maintaining their mutual visibility), it is 1-searchable according to the same reason as above. However, there is some 1-searchable room which is not searchable by two guards. The example is shown in [74].

The work in [73, 93, 108] also discussed some other search problems in a room.

1.3.5 Graph-based solutions to polygon search problems

In the previous discussions of the polygon search problems, including the street and room search problems, the techniques used are basically the *characterizations* of polygons, in which a set of patterns in a polygon are identified. Depending on whether a polygon contains these patterns or not, we then decide its searchability. Usually, the proofs of these results are quite lengthy and involved since one has to go through each of the patterns completely before the final conclusion is drawn.

In this section, we will discuss how to deal with polygon search problems using another line of approach. The new approach could be regarded as the application of the visibility polygons discussed in Section 1.2.

Instead of enumerating possible patterns, we visualize a diagram which consists of the information, for each point on the boundary of the polygon, in the visibility polygons. We then identify the key points on this diagram and use them to conduct our polygon search process.

LaValle *et al.* [67, 68] discussed the search problem using a 1-searcher on the boundary of a given polygon. For a given polygon P, the searcher moves along ∂P and uses a flashlight to separate the cleared and contaminated portions of the polygon.

For points $p, q \in \partial P$, as defined in [67, 68], p is visible to q if every interior point of the line segment $\overline{pq}$ lie in $P - \partial P$. Thus, no two points on the same edge of P are mutually visible.

A *configuration* is defined to be a pair $\langle p, q \rangle$ of points $p, q \in \partial P$, and the space of all configurations X is defined as: $X = \partial P \times \partial P = \{\langle p, q \rangle | p, q \in \partial P\}$.

There are three types of configurations we need to distinguish.

The *diagonal configurations*, denoted as $X_d \subset X$, that contain the pairs $\langle p, q \rangle$ such that p and q lie on the same edge of ∂P.

The *feasible configurations*, denoted as $X_v \subset X$, that contain the pairs $\langle p, q \rangle$ such that p and q are mutually visible.

The *non-feasible configurations*, denoted as $X_n = X - X_d - X_v$, that contain the pairs $\langle p, q\rangle$ that do not lie on the same edge and are not mutually visible.

Note that the definition of mutual visibility here between two points is a little different from our definition in Section 1.2, in that if a beam touches a reflex vertex, it cannot travel beyond that point. In our definition of the mutual visibility, we have $X_d \subset X_v$. We will see this in later chapters.

Intuitively, for any $\langle p, q\rangle \in X_v$, p can represent the position of the searcher while q can represent the point (the beam head) illuminated by her flashlight. We now give the definition of a visibility obstruction diagram.

Definition The *Visibility Obstruction Diagram* (VOD) for a polygon P is defined as a diagram consisting of three partitions of X, $\langle X_d, X_v, X_n\rangle$, where X_d, X_v, and X_n are defined as above.

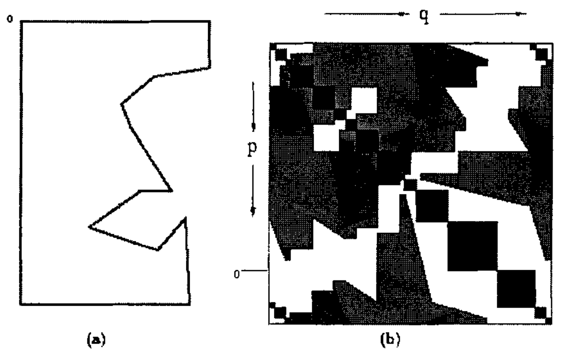


Figure 1.11: A simple polygon and its corresponding visibility obstruction diagram.

Figure 1.11 shows an example of a polygon and its corresponding VOD. In the diagram, the black area along the diagonal represents X_d, the gray (shaded) area represents X_n, and the white area represents X_v.

A feasible search schedule may consist of the moves (the vertical moves of the searcher and the horizontal moves of the flashlight) inside the white areas in the diagram. Occasionally, the move can go across a gray area from right to left, representing a recontamination (we will see more on this in later discussions). In order to simplify this construction, several definitions were introduced [67, 68].

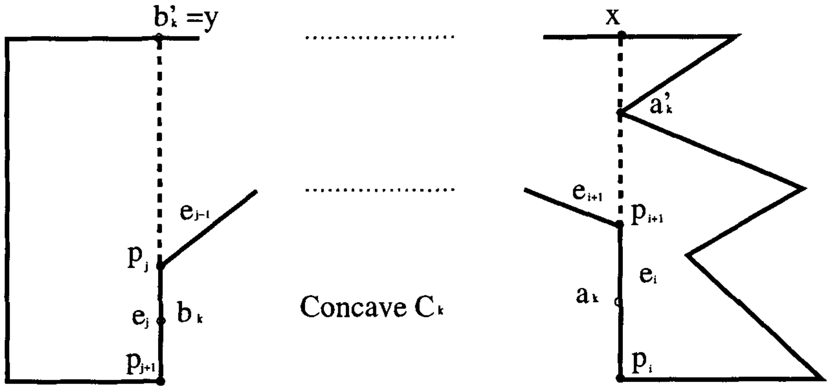


Figure 1.12: A concave region.

A maximal subinterval of ∂P of the form $C = (p_i, p_{j+1})$, in which all of the vertices $p_{i+1}, p_{i+2}, \cdots, p_j$ are reflex vertices, form a *concave region.* Consider the k-th concave region $C_k = (p_i, p_{j+1})$ of P from p_0 in the clockwise direction. Let e_i be the edge between the vertices p_i and p_{i+1} and e_j be the edge between the vertices p_j and p_{j+1}. Define the *shelter* of C_k to be the two points a_k and b_k, the midpoints of the edges of e_i, e_j, respectively. Shoot a ray starting at a_k through p_{i+1} and let x be the point of ∂P where the ray leaves P for the first time. Define the *threshold* of a_k, denoted by a_k', to be the point in $\overline{a_k x} \cap \partial P$, which is the nearest from a_k in the counter-clockwise direction along the boundary. The *threshold* of b_k can be defined similarly. See Figure 1.12 for an example for a concave region.

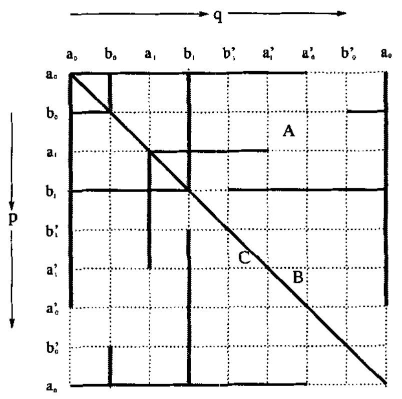

q
p
a₀ b₀ a₁ b₁ b'₁ a'₁ a'₀ b'₀ a₀
a₀ b₀ a₁ b₁ b'₁ a'₁ a'₀ b'₀ a₀
A
B
C

Figure 1.13: The skeletal obstruction diagram corresponding to Figure 1.11.

Now we identify these shelter and threshold points in the VOD diagram. We finally obtain the *skeletal obstruction diagram* (SOD), which represents the *skeleton* of the VOD, drawn over the square grids formed by the shelter and threshold points. For example, for the VOD shown in Figure 1.11, its corresponding SOD is shown in Figure 1.13. It is proven in [67, 68] that this representation is topologically equivalent to the VOD in terms of the searchability of a polygon by a 1-searcher.

In order to search in SOD effectively, a graph G_P is constructed as follows. The vertex set $V(G_P)$ is composed of three types vertices: (1) the non-diagonal vertices, one for each square that does not intersect the diagonal (such as the one labeled by A in Figure 1.13); (2) the starting vertices, one for each half-square, a triangle immediately above the diagonal (such as the one labeled by B in Figure 1.13); and (3) the goal vertices, one for each half-square, a triangle immediately below the diagonal (such as the one labeled by C in Figure 1.13). The edge set of the graph is defined as the maximum subset of $V(G_P) \times V(G_P)$ satisfying five rules. See [67, 68] for the details of these rules. We do not detail them here for the sake of space.

The whole algorithm for deciding whether a polygon is 1-searchable and if so, constructing a search schedule, can be basically described as follows. For the given polygon, determine the shelter and threshold points and then construct the SOD and the graph G_P. Finally use a breadth-first search (BFS) to find a path in the graph from a starting vertex to a goal vertex. If no such path exists, we say that the polygon is not 1-searchable. The following theorem is established.

Theorem 1.3.6 *Given a simple polygon P with n vertices and m concave regions, there is an algorithm that decides whether it can be searched by a 1-searcher, and if so, outputs a search schedule in time $O(n + m \log n + m^2)$.*

Another graph-based solution was due to Guibas *et al.* [55]. They studied how to determine the minimum number of searchers for both simple polygons and polygons with holes and how to construct an *information graph* in order to find the search schedule for a polygon. Note that the searchers in their work have a 360^o visibility range. The following two lemmas are about the minimum number of searchers.

Theorem 1.3.7 *Let $ps(P)$ denote the minimum number of searchers that are needed to search a polygon. For a simple polygon P with n edges, $ps(P) = O(\log n)$, and there exists some polygon P' with n edges such that $ps(P') = \Omega(\log n)$. These two statements together imply $ps(P) = \Theta(\log n)$.*

This result is consistent with the one in Theorem 1.3.1 and also with the one we are going to introduce in Section 1.3.8. The technique used is similar to the one employed in Theorem 1.3.1.

Theorem 1.3.8 *Let $ps(P)$ denote the minimum number of searchers that are needed to search a polygon. For a polygon P with n edges and h holes, $ps(P) = O(\sqrt{h} + \log n)$, and there exists some polygon P' with n edges and h holes such that $ps(P') = \Omega(\sqrt{h} + \log n)$. These two statements together imply $ps(P) = \Theta(\sqrt{h} + \log n)$.*

It was observed [55] that in a polygon P, the boundary of a visibility polygon from a boundary point q generally alternates between being part of the boundary and crossing the interior of the polygon. Let each edge of a visibility polygon that is in the interior of the polygon be a *gap edge.* We assign a label "1" to it if the subpolygon (that is on the different side from the visibility polygon) is contaminated and assign a label "0" to it, otherwise. Let $B(q)$ denote the binary vector of gap edge labels, with one label for each gap edge in q's visibility polygon. Thus the pair $\langle q, B(q)\rangle$ uniquely characterizes the information state when the searcher is at q. Then we partition the given polygon using a collection of convex regions. Now construct a graph G, each vertex of which corresponds to a convex region and each edge of which is defined if the boundaries of two regions have a one-dimensional intersection. A directed *information graph*, G_I, can then be derived from G. For each vertex in G, a set of vertices are included in G_I, one for each possible labeling of the gap edges. The edge between two vertices in G_I depends on when a gap edge disappears, a gap edge appears, two or more gap edges merge, or a gap edge splits into two or more. The search process starts at a vertex in G_I where $B(q) = 11\cdots1$ to a goal vertex where $B(q) = 00\cdots0$. The paper showed several computed examples, demonstrating the feasibility of the approach. But the formal analysis of the complexity was not conducted.

Simov *et al.* [101] used the same graph-based idea to solve the polygon search problem. It is claimed that the algorithm needs $O(n^3)$ time to guarantee that a successful search schedule would be found.

Simov *et al.* [99] considered the polygon search problem using two 1-searchers (not necessarily bound to the boundary of a given polygon). The solution to the problem was also graph-based. They considered the shelters in a given polygon, as discussed above. They also considered a *bitangent.* A bitangent is defined as follows. Let $c, d, c', d', x \in P$ be collinear points. Let c and d be mutually visible vertices of ∂P and x be an interior point of

$\overline{cd}$. If there exist points $c^{'}$ and $d^{'}$ such that the pairs $(c, c^{'})$ and $(d, d^{'})$ form gap edges relative to x, we say that c and d define a *bitangent*, and we call c and d the bitangent points and $c^{'}$ and $d^{'}$ the bitangent projections. The *shelter* and *bitangent* configurations were introduced. They form the set of the *b-configuration.* Let $\tau_1 = \langle p_1, q_1 \rangle$ and $\tau_2 = \langle p_2, q_2 \rangle$ be a pair of b-configurations and let b be the corresponding contamination string (defined similarly to $B(q)$ as discussed above). Basically, $\tau_1 = \langle p_1, q_1 \rangle$ and $\tau_2 = \langle p_2, q_2 \rangle$ decide the statuses of the two searchers. Then a *b-state* is defined to be (τ_1, τ_2, b). The *b-space* is defined as the set of all b-states. Now a *directed information graph* can be composed as follows. Its vertex set consists of all b-states. Two vertices (b-states) are connected if there is an elementary move from one to the other. The search schedule is finally obtained by conducting a breadth-first search (BFS) in this graph. A set of elementary moves were illustrated and discussed in [99].

Another useful diagram, called the cylindrical *boundary visibility diagram* (BVM), was introduced by Suzuki *et al.* [105] for an arbitrary search schedule S of an ∞-searcher on the following observations. The boundary of a polygon is topologically equivalent to a circle. The "state" of such a circle, which is a function of time t, consists of information on the current locations of *pockets* (similar to the caves defined in Figure 1.10), if any, and whether or not they are contaminated or not. The cylindrical BVM arranges these circles next to each other, from left to right, in the order of time t $(0 \leq t \leq t_0)$, where t_0 is the time the polygon is cleared. We distinguish three sets of points in BVM at time t: (1) the set of all (t, p) $(p \in \partial P)$ such that p belongs to a contaminated pocket; (2) the set of all (t, p) such that p belongs to a cleared pocket at time t; and (3) the set of all (t, p) such that p does not belong to any pocket. As the ∞-searcher traverses, the pockets may emerge, disappear, merge and split. The BVM diagram describes when these events happen. The major contribution in the paper is stated in the following theorem [105].

Theorem 1.3.9 *A polygon P can be cleared, using a boundary schedule, by an ∞-searcher, if only if it can be cleared, using a boundary schedule, by a 1-searcher.*

For other work investigating the 1-searcher polygon searcher problem, see [91, 92, 109]. They will also be discussed in the following.

1.3.6 The chain of k-guard problem

The polygon search problem by a chain of k-guard was discussed by Efrat *et al.* [41]. The search process is subject to the restriction that the first and the k-th searchers always move on the boundary of a given polygon while searcher i, $1 < i < k$, moves in the interior and maintains the visibility with its neighbors: searchers $i-1$ and $i+1$.

They attempted to compute a minimum number r^* of searchers needed to sweep a polygon with n vertices under the chain of k-guard mechanism. A new concept *link diagram* was introduced and, as claimed, was the first of its kind in the literature. A link diagram provides another tool to analyze the polygon search problem. Compared with the graph-based solutions as discussed above, the link diagram captures other geometric information hidden inside a given polygon.

Another version of k-guarding problem was proposed and studied by Belleville *et al.* [9]. A polygon is called k-guardable if it is possible to find a collection G of points in the interior of the edges of a given polygon P such that every point in P is visible from at least k elements of G. Due to the space limitation, we will not go into details. Interested readers may refer to the paper for further information.

1.3.7 The k-searcher problem

To the best of our knowledge, although there are some results obtained for ∞-searcher [105, 109], there is not much progress towards $k \geq 3$ for a k-searcher. In this section, we discuss some work related to the case where $k = 1$ or $k = 2$.

Park *et al.* [90] discussed the 1-searchability of a polygon. Basically the techniques used are similar to the ones in [89]. We will not repeat it here. Instead, we introduce another paper by Tan.

Based on a case-by-case analysis, the following theorem was established by Tan [109].

Theorem 1.3.10 *A polygon P is 1-searchable if and only if none of the four configurations, as shown in Figure 1.14 (where the dashed lines represent the rays from reflex vertices), appears in the polygon.*

Due to the space limitation, we will not detail the proof here since it is quite lengthy and tedious. Note that in [109], the 1-searcher may go off the boundary of a polygon. The paper also established that if a polygon is searchable by an ∞-searcher, then it is also searchable by a 2-searcher. This work could be regarded as complementary to the work reported in [105].

There is some other work discussing the k-searcher problem. Interested readers may refer to [72].

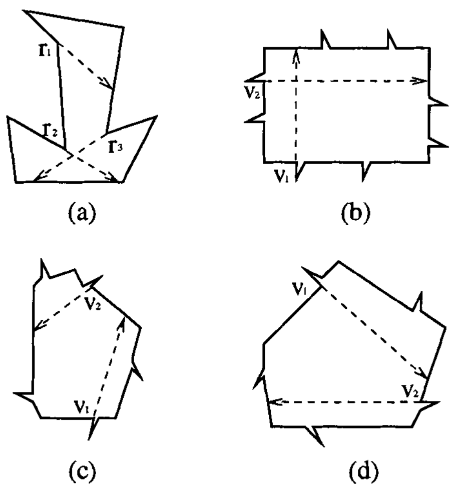


Figure 1.14: Four configurations that make a polygon not 1-searchable.

1.3.8 Polygon metrics related to polygon search problems

It is believed that different polygon metrics will help better understand polygon search problems. In this section, we discuss some results in this direction.

Yamashita *et al.* [115] explored the different metrics of a polygon. Denote by $ps(P)$ the number of searchers necessary and sufficient to search a given polygon P, with n vertices and r reflex vertices. There are another two measures of the shape complexity of P that are of our interest. A finite set G of points in P is called a *guard set* of P if every point $x \in P$ is visible from some point $y \in G$. We define g as the minimum size $|G|$ among all guard sets G of P. The definition of the bushiness b of P is based on a triangulation of P. It is known that a triangulation of a polygon with n vertices has $n-2$ triangles and $n-3$ non-intersecting diagonals [12, 81]. The dual of a triangulation is a tree T having a vertex for each triangle and an edge between those vertices that correspond to the two triangles that share a common diagonal. A triangulation is said to be *thin* if its dual tree has the smallest number of degree-three vertices among all triangulations of P. The *bushiness* of P is then defined to be the number of degree-three vertices in the dual tree of a thin triangulation of P.

Given the polygon P and an edge e, the *chasing problem* with respect to e is the problem of discovering all intruders in room P without letting any of them escape from the entrance e. The *shooing problem* with respect to e is the problem of clearing room P with entrance e

through which a new intruder may enter into P at any time. Thus, with those two problems, the *corridor search problem* (also discussed above) for P with respect to the entrance e and the exit e' is the problem of designing a search schedule for P which is both a chasing schedule with respect to e and a shooing schedule with respect to e'.

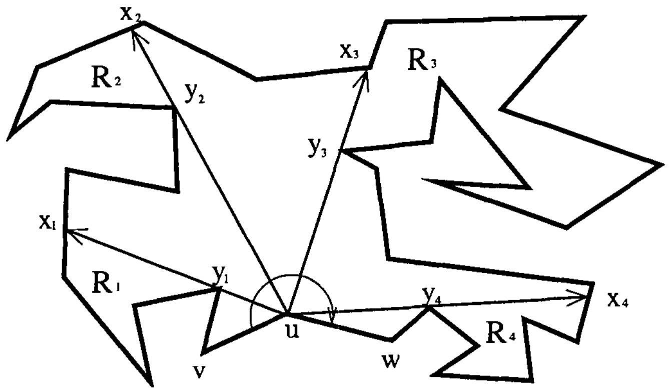


Figure 1.15: An example of one-way sweep strategy.

The method for constructing a search schedule is to decompose the original problem into smaller chasing, shooing, and corridor search problems until each smaller problem becomes tractable. This process is formalized in an algorithm called *one-way sweep strategy* (OWSS). Figure 1.15 shows an example. For the details of the algorithm, see [115].

The work also related the polygon search problem to the graph search problem [94]. Denote by $es(G)$ the *graph search number* of G, which is the minimum number of searchers for searching a graph [94]. The following lemma regarding the dual tree of a triangulation of a polygon was established [115].

Lemma 1.3.11 *Let T be the dual tree of a triangulation of a simple polygon P. Then $ps(P) \leq es(T)$.*

With the help from [77] showing that if T is a tree with $m \geq 2$ vertices, then $es(T) \leq 1 + \lfloor \log_3 (m-1) \rfloor$, the following two theorems regarding the number of vertices and the bushiness of P were proven [115].

Theorem 1.3.12 *For any simple polygon P with n vertices, $ps(P) \leq 1 + \lfloor \log_3 (n-1) \rfloor$.*

The theorem that relates $ps(P)$ to the bushiness of a polygon was also proven [115].

Theorem 1.3.13 *For any simple polygon P with bushiness b, $ps(P) \leq 1 + \lfloor \log_3 (2b+1) \rfloor$*

By constructing a search schedule based on the recursive applications of OWSS, the following theorem related $ps(P)$ to the number of reflex vertices of P in the paper.

Theorem 1.3.14 *For any simple polygon P with $r \geq 1$ reflex vertices, $ps(P) \leq 1 + \lfloor \log_3 r \rfloor$*

The following theorem related the size of a guard set with $ps(P)$.

Theorem 1.3.15 *Let P be a simple polygon having a guard set of size g. Then $ps(P) \leq 2 + \lfloor \log_2 g \rfloor$*

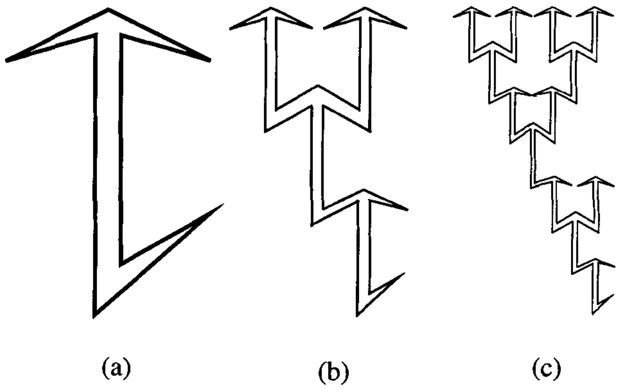


Figure 1.16: Examples that provide the lower bounds for $ps(P)$.

Using the configurations shown in Figure1.16, the work finally showed us that there are some simple polygons that do provide the lower bounds on the polygon search number as discussed in the above four theorems, as stated in the following theorem.

Theorem 1.3.16 *For any natural number $s \geq 2$, there is a simple polygon P satisfying $ps(P) = \log_3 (n+1) = log_3(2r+3) = log_3(2b+1) + 1 = log_3(2g-1) + 1 = s$, where n, r, b, and g are the numbers of vertices, the number of reflex vertices, the bushiness, and the size of a minimum guard set of P, respectively.*

The work in [114] was an extended abstract of the work we are introducing here. Also the work [107] discussed the problem of how to represent $ps(P)$ in terms of the bushiness of a polygon. Given the guards' positions *à priori*, Cheong *et al.* [28] discussed how to compute a guarded region.

1.3.9 Other related polygon search problems

In the following, we discuss some other interesting topics related to the polygon search problems in the literature.

The Robber Route Problem

A problem called the *Robber Route* was introduced by Ntafos [78]. The problem is defined as follows. Consider a set of T points and a set E of edges of a polygon P. The goal is to find a closed walk W starting and ending at a point x such that every point on any edge of E is visible from a point in W while W is not visible from any point in T. The elements in T are threats that we desire to avoid while the elements of E are the sights that we might want to see.

The Zookeeper and Safari Route Problem

Chin [30] proposed and studied another variation of the Robber Route Problem. The task is to find a route in a polygon P containing a set of sites S and a set of sites T. The sites in S are sights that must be visible from at least one point on the route while the sites in T are threats that must never be visible from any point along the route. If the threats are represented by polygons within the given polygon P, the problem becomes the *Zookeeper Route Problem.* By restricting these polygons to be attached to the edges of the outer polygon, the work designed an algorithm for finding the shortest path for this purpose and showed that it is unique. In general, the problem is NP-hard. See [50] for NP-hardness.

The *Safari Route problem* is the same as the Zookeeper Route Problem, except that one is allowed to enter the threats. It was proven by Ntafos [79], that in general this problem is also NP-hard. But if the sites were allowed to attach to the edges of the outer polygon, the problem could be solved in $O(n^3)$ time.

The Lazy Guard Problem

Colley *et al.* [32] studied the so-called *Lazy Guard Problem.* The problem is defined as follows. Given a polygon P, choose a minimal number of stations (represented by points) in the polygon such that a mobile searcher who visits all stations will guard the entire polygon. The problem has its counterpart in real applications. For example, a searcher is inherently lazy in the sense that she may not patrol as often and thoroughly as her supervisor wishes.

To ensure the completeness of the patrol, the supervisor will install a set of *check-in* stations such that the searcher has to physically visit them on a regular basis.

It was proven that an optimal placement of stations for lazy guarding a simple polygon can be found in linear time. It was also shown that the Lazy Guard Problem is NP-complete for polygons with holes. See [50] for NP-completeness.

1.3.10 Related topics

The polygon search problems are of interest to other research communities. The pursuit-evasion problem in graphs is to search a graph by a group of searchers. See [10, 17, 63, 76, 77, 94] for some work related to the problem.

Many variations of the polygon search problem also play an important role in robotics. Many of the results we have discussed above were used in solving problems related to plane explorations. See [3, 8, 13, 37, 53, 64, 65, 66, 95, 96] for some results in this direction.

There are also some other papers discussing how to explore unknown environments in the literature [2, 6, 8, 39, 40, 59, 60, 87, 88].

For the work related to the *Floodlight Illumination Problem*, see [1, 20, 34, 46, 47, 48, 83, 102].

For other work related to the topics, such as visibility, triangulations, visibility graphs, etc., see [4, 5, 7, 11, 16, 18, 19, 21, 22, 25, 27, 35, 38, 42, 43, 44, 45, 49, 52, 61, 62, 69, 70, 71, 82, 84, 85, 86, 97, 103, 104, 111, 112].

1.3.11 *LR*-visible polygons

In our later discussions, we will use a class of *LR*-visible polygons. We briefly discuss them here.

As defined in [36], a polygon P is LR-visible if there exists a pair of points s and t on ∂P such that $\partial P_{cw}(s,t)$ and $\partial P_{ccw}(s,t)$ are weakly visible from each other. Also in the same paper, if a polygon is LR-visible, a linear-time algorithm is shown to determine all possible pairs of boundary chains (A_i, B_i), $i = 0, 1, \cdots, m$, such that for any $s \in A_i$ and any $t \in B_i$, P is LR-visible with respect to (s,t). As a preparatory step, there is another algorithm that calculates all of the non-redundant components in linear time. If the polygon is not LR-visible, this algorithm terminates prematurely.

Figure 1.17 shows such an example. We change the notation in the figure in order to be

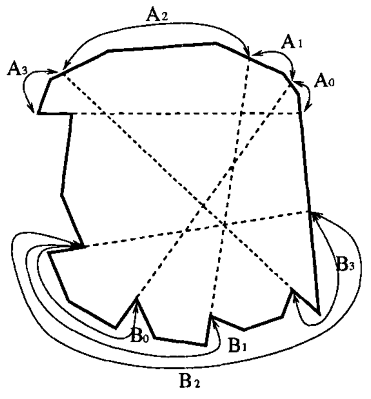


Figure 1.17: An *LR*-visible polygon adopted from [36].

consistent with the one used in our discussions. It is easy to check that the polygon is *LR*-visible with respect to any pairs of points from (A_0, B_0), (A_1, B_1), (A_2, B_2), and (A_3, B_3). Note that as discussed in [36], the chain pairs can be output either with all A_i's disjoint or all B_i's disjoint. For example, in Figure 1.17 all the chains A_i's are disjoint. We will use this result later.

Another concept that will be used in our discussions is the *shortest path tree.* A shortest path between two vertices u and v on ∂P is a sequence of maximal line segments connecting u and v that lie entirely within P and has the minimum Euclidean distance. Shortest paths are unique [36]. The *shortest path tree* from a vertex v on ∂P, denoted by $SPT(v)$, is the union of the shortest paths from v to u, where u is a vertex of P. For a simple polygon, the shortest path tree from a vertex v can be computed in linear time [54].

1.4 Our work

The focus of our work in this thesis is to improve upon the previous work while attempting to solve new problems.

In our work, we first further investigate the room search problem. The problem has been attempted before. Considering that its counterpart, the street search problem, can be checked in linear time for searchability, we are seeking the same optimality in the room searchability problem. In addition, being not satisfied with the previous involved and complex analysis, we attempt, with the help of previous results, to simplify the solution to the room searchability problem. Our approach solves the room search problem by two searchers as well as by a 1-searcher in a uniform way. We also solve a new problem related to the room search problem - finding all the possible doors of a polygon such that the resultant rooms are searchable.

For the polygon search problem with a 1-searcher, the previous characterization, as discussed in Section 1.2, is quite tedious and complex. We desire to find a concise solution to the problem. We also relate the searchability problem to the *LR*-visibility problem of a polygon, with the hope that the latter could help us improve the time complexity of the former. We also explore the relationship between the polygon search problem and the room search problem by a 1-searcher. We will present our results in these problems.

The polygon search problem by two 1-searchers has been studied before. In our work, we focus on a restricted version of the problem. We study the search problem by two 1-searchers in the polygon with one hole. We consider three cases: (1) both searchers move on the inner boundary of the polygon; (2) both searchers move on the outer boundary; and (3) one searcher moves on the inner boundary and the other moves on the outer boundary. We provide an algorithmic graph-based solution to the problem.

The remaining part of the thesis is structured as follows. In the next chapter, we prepare necessary notation for the later chapters. We introduce and discuss the *visibility obstruction diagram* (VOD) and the *skeletal visibility obstruction diagram* (SVOD) of a given polygon, which is the basis of our work in this thesis.

In Chapter 3, we construct the VOD and SVOD of a given room and use it to help us analyze the searchability problem of a room by two guards. We also show that a new problem of finding all doors given a polygon can be solved optimally. The solutions we obtain can be easily extended to the room search problem by a boundary 1-searcher.

Chapter 4 revisits the polygon search problem by a 1-searcher. We will see how the revision of the VOD and SVOD of a polygon helps reveal more of the inherent nature of the problem. We also relate the polygon's searchability (by a 1-searcher) problem with its *LR*-visibility. It might appear that a 1-searchable polygon is a searchable room, i.e., we can find a door on the boundary such that the resultant room is 1-searchable. However, our result shows that this is not always possible.

In Chapter 5 we propose a new problem of searching a polygon with a polygonal "hole" inside. We use two 1-searchers for this task. We explore several variants of the problem. Again, we make use of the extended visibility obstruction diagram to guide our analysis.

Finally, in Chapter 6, we summarize our contributions in this thesis and also indicate some further extensions and future work on the basis of our results here.

Chapter 2

Preparations

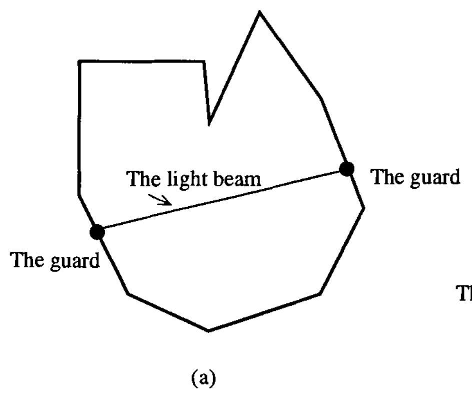
2.1 Introduction

Visibility between any two boundary points plays an important role in various polygon search problems. Given a polygon P, from a point $p \in \partial P$, the visibility polygon $V(p)$ consists of those points of P that are visible from p (see Section 1.2 for details.). Let us take the 1-searcher polygon search problem as an example. If the searcher is standing at p, we immediately know that the beam head of her flashlight can only be falling on $V(p) \cap \partial P$. For the same reason, for the two-guard search problem, no matter whether the polygon to be searched is a street or a room, if one guard is standing at p, the other guard must be on $V(p) \cap \partial P$ since otherwise the mutual visibility requirement is violated.

The visibility polygons of two different boundary points may overlap. If we consider the visibility polygons for all the boundary points for a given polygon, the collective information hidden in them *might* help us decide whether the polygon (under different search models) is searchable or not. Actually the answer to this is affirmative, as will be seen in what follows.

2.2 Two search models

We consider two search models in this thesis. The first one is the two-guard model. In this model, the two guards move on the boundary of a polygon. It is required that the two guards be always mutually visible throughout a search process. The visibility range of the two guards in this search model is restricted to the line segment connecting them. (For



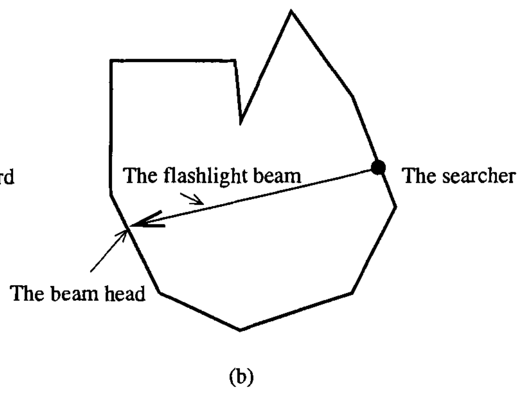


Figure 2.1: Two search models

convenience, we imagine this line segment to be a light beam emanating from a flashlight from one of the two guards.) An intruder is caught if he is hit by the light beam. Figure 2.1 (a) illustrates this. We use boundary points to represent the two guards.

The second model we consider is the boundary 1-searcher model. In this model, there is a searcher with a flashlight, who always moves on the boundary of a polygon while aiming her flashlight in any direction she wants. The head of the beam (ray) from her flashlight is the point where the beam and the boundary of the polygon intersect before the beam leaves the polygon for the first time. The visibility range of the searcher is restricted to the beam from her flashlight. An intruder is caught if he is hit by the beam. We illustrate this model in Figure 2.1 (b), where the searcher is represented by a point on the boundary.

Some previous work has investigated different polygon search problems under these two models. See Chapter 1 for a review of them. We will come back to these two search models when we discuss different search problems in later chapters.

2.3 Visibility space and visibility obstruction diagram

Given a polygon P, we define a *configuration* (introduced by LaValle *et al.* [67, 68]) to be an element in $\partial P \times \partial P$. For a configuration $\langle p, q\rangle$, where $p, q \in \partial P$, if p and q are mutually visible, we say that $\langle p, q\rangle$ is a *visible* configuration. Otherwise, we say that it is an *invisible* configuration.

The *visibility space* of P is defined to be the set of all configurations for P. It is obvious that, because $\overline{pq} \in P$ if and only if $\overline{qp} \in P$, the visibility space is symmetric in the sense that if the configuration $\langle p, q \rangle$ is a(n) (in)visible configuration, so is the configuration $\langle q, p \rangle$. See discussions in Section 1.3.5.

Following what LaValle *et al.* [67, 68] proposed, we visualize the visibility space using a diagram, which is represented by a unit square in this chapter. Starting from the origin vertex p_0 of P, we map ∂P monotonically into the sides of this diagram. Note that the definition of visibility in [67, 68] is different from the one adopted here. In their definition, $\langle p, q \rangle$ is a visible configuration if and only if $\overline{pq}$ is in the *interior* of P. Thus, two points on the same edge are not mutually visible. There are black squares along the diagonal, representing mutual invisibility as shown in Figure 1.11 in Chapter 1.

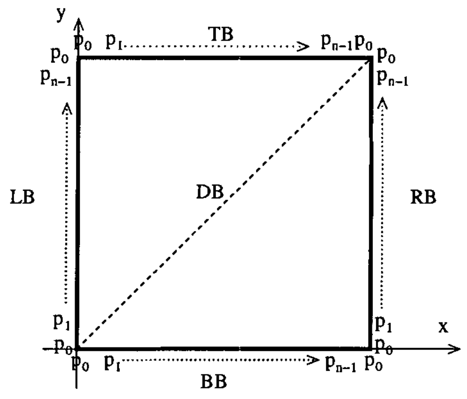


Figure 2.2: A representation of the visibility space.

Figure 2.2 is a visualization of the visibility space. The vertical axis of the diagram represents, from bottom to top, one traversal of ∂P in the clockwise direction while the horizontal axis also represents, from left to right, one clockwise traversal of ∂P. We label the relative positions of vertices on the four sides in their respective orders. For convenience, we call the top side of the diagram the *top boundary* (*TB*, for short), the right side the *right boundary* (*RB*, for short), the bottom side the *bottom boundary* (*BB*, for short), the left side the *left boundary* (*LB*, for short), and the diagonal line (running from the bottom-left corner to the top-right corner) the *diagonal boundary* (*DB*, for short). We also show the coordinate system for the diagram in the figure. For any point (x, y) (an ordered pair) in the diagram, x is projected on *TB* and *BB*, and y is projected on *LB* and *RB*.

We now construct the *visibility obstruction diagram* (VOD) [67, 68] [1] to represent the

[1]In the VOD as defined by LaValle *et al.*, the values in the horizontal axis increase in the clockwise

visibility relationship among the boundary points of a given polygon in the visibility space. The type (visible or invisible) of each configuration $\langle p,q\rangle$ of a polygon P at point (p,q) is indicated in *VOD*(P) by a different color. The points in *VOD*(P) representing invisible configurations are shaded gray while those representing visible ones are left white.

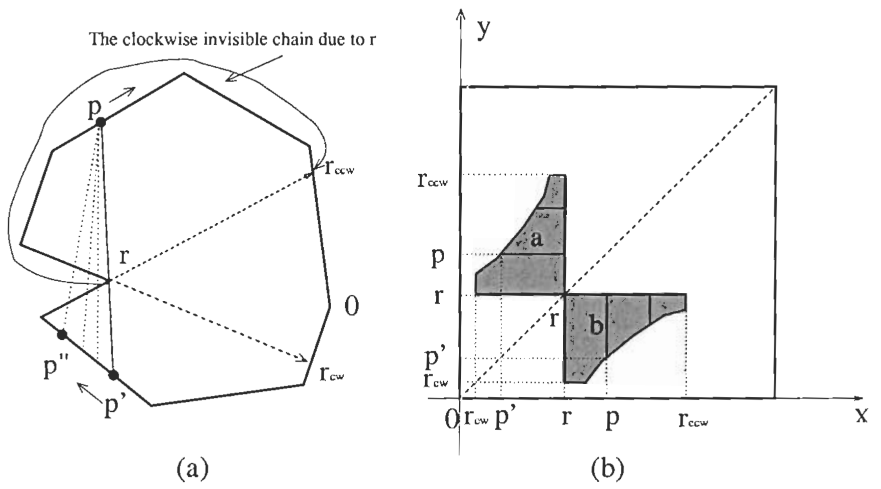


Figure 2.3: The relationship between invisible configurations and an invisible chain.

We consider the properties of the VOD for a polygon P. We will show how the invisible configurations are related to the invisible chains due to reflex vertices in P and how this relationship is presented in *VOD*(P).

We first discuss the clockwise invisible chain caused by a reflex vertex on ∂P. In Figure 2.3 (a), we show a reflex vertex r and the clockwise invisible chain $\partial P_{cw}(r, r_{ccw})$ due to it. Now select a point $p \in \partial P_{cw}(r, r_{ccw})$. We shoot a ray in the direction from p to r. Suppose that the beam head of the ray is at p', as shown in the figure. It is obvious that for any point p'' different from p' on $\partial P_{cw}(p', r)$, $\langle p'', p\rangle$ and $\langle p, p''\rangle$ are invisible configurations. If we project them to the VOD, it is easy to see that for p, the gray points corresponding to the invisible configurations $\langle p'', p\rangle$ form a horizontal gray line (as marked by a in Figure 2.3 (b)) starting from p' and ending at r, and the gray points corresponding to the invisible configurations $\langle p, p''\rangle$ form a vertical gray line (as marked by b in the Figure 2.3 (b)) starting from p' and ending at r.

direction of the vertices of a polygon, while the vertical axis represents the counter-clockwise direction of them. We adopt the standard coordinate system so that both axes represent the clockwise direction of vertices. However, we will retain the name VOD.

Now consider all the points on $\partial P_{cw}(r, r_{ccw})$. They generate a set of vertical gray lines and a set of horizontal gray lines (generating two gray areas). This is shown in Figure 2.3 (b). Also as point p approaches r_{ccw} in the clockwise direction, the intersection point p' approaches r in the clockwise direction as well, and the length of the corresponding gray line ($=|\partial P_{cw}(p', r)|$) gradually (actually monotonically) becomes shorter. The two gray areas extend from r on DB upward and rightward as p approaches r_{ccw}.

It is easy to see from the above discussions that the gray areas caused by a reflex vertex is symmetric with respect to DB in the VOD. If we draw the gray areas due to all the reflex vertices, some of them may overlap.

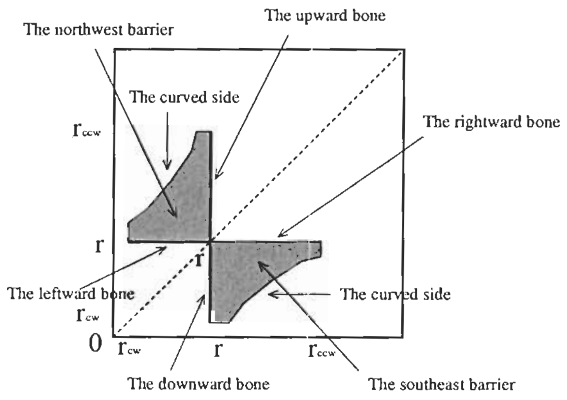


Figure 2.4: A barrier and its bones.

The two gray areas due to r shown in Figure 2.3 (b) are represented in Figure 2.4 with some annotations. As shown in the figure, we call the gray area due to r above (below, resp.) DB the *northwest barrier* (*southeast barrier*, resp.) due to r and denote it as $NW(r)$ ($SE(r)$, resp.). We call the bottom flat side of $NW(r)$ the *leftward bone* to indicate its extension direction. It is obvious that the length of the leftward bone is $|\partial P_{ccw}(r, r_{cw})|$. We call the barrier's right flat side the *upward bone*, whose length is $|\partial P_{cw}(r, r_{ccw})|$. Similarly for $SE(r)$, we call its top flat side the *rightward bone* and its left flat side the *downward bone*. We also indicate the curved side of each barrier.

For example, we show a polygon and its corresponding VOD in Figure 2.5. The two barriers marked by the thick lines in the figure are due to reflex vertex 4. In the figure, we also mark (using thick arrows) the four extension directions of the barriers. The remaining gray areas in the figure are due to other reflex vertices, i.e., vertices 8 and 10.

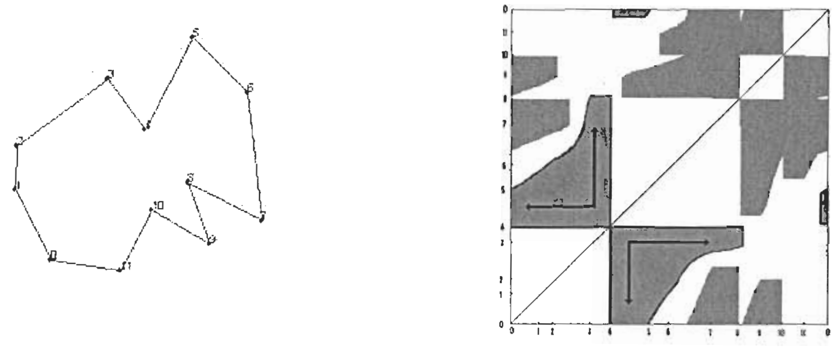

(a) (b)

Figure 2.5: A polygon and its corresponding VOD.

Proposition 2.3.1 *Every reflex vertex r of P gives rise to two barriers. The two barriers are symmetric with respect to DB. Barrier $NW(r)$ extends upward and leftward from r on DB in VOD(P) and stops at r_{ccw} and r_{cw}, respectively, while barrier $SE(r)$ extends downward and rightward from r on DB and stops at r_{cw} and r_{ccw}, respectively.*

Note that $VOD(P)$ wraps around horizontally and vertically. Consider barrier $SE(r)$, such as $SE(4)$ in Figure 2.5, for example. If the counter-clockwise (clockwise, resp.) invisible chain due to r contains p_0 (which is the vertex 0 in the example figure), because BB (LB, resp.) corresponds to point p_0, SE (NW, resp.) extends to BB (RB, resp.), and continues at the position r on TB (LB, resp.) downward (rightward, resp.) and stops at r_{cw} (r_{ccw}, resp.). Barrier $NW(r)$ is treated similarly. This means that the number of barriers in the VOD due to a reflex vertex is still two.

The barriers due to different reflex vertices might intersect. For instance, in Figure 2.5, the leftward extension of barrier $NW(10)$ intersects the upward extension of barrier $NW(8)$.

Another fact is that the invisible chains from a reflex vertex are continuous. Thus, the corresponding barriers in the VOD are also continuous, as shown in their construction. That is, there are no white areas inside a barrier. On the other hand, a white area might be surrounded by barriers due to different reflex vertices in the VOD. But it should be noted that it is not inside any barrier.

2.4 Skeletal visibility obstruction digram

From the above discussions we know that the points inside a barrier due to a reflex vertex in the VOD of polygon P represent an infinite set of invisible configurations.

Consider the first search model of two guards. With a configuration $\langle x, y\rangle$, if we use x to represent the current position of one guard on ∂P while using y to represent the current position of the other under the mutual visibility requirement, we know that $\langle x, y\rangle$ must be a visible configuration, which means that corresponding point in the VOD should not be in a barrier.

As for the second search model of a 1-searcher, if we use y to represent the current position of the searcher on ∂P and x to represent the beam head of her flashlight on ∂P, the configuration $\langle x, y\rangle$ must be a visible one. We will discuss the second search model in more detail in Chapter 4.

Thus, within a barrier due to a reflex vertex in the VOD, one point is equivalent to another in terms of the invisible configurations they represent. In other words, the key information we can make use of to conduct the polygon searchability studies is contained in the *topology* of the VOD, rather than the actual shapes of barriers.

The invisible configurations are caused by the invisible chains due to reflex vertices. Thus the length of each invisible chain is important to determine to what extent the reflex vertices pose difficulties to polygon searches.

In order to facilitate our polygon searchability analysis, we collapse each barrier toward its two bones (representing the two invisible chains), extracting the topological information from the VOD. We obtain, for each extension, a *bone segment* (or simply a *bone*). (We abuse the notation a little here.) If a bone is generated from a barrier due to a reflex vertex r, we also say that the bone is *due to r*.

The length of each bone is equal to that of the corresponding invisible chain. We call the diagram thus obtained the *skeletal visibility obstruction diagram (SVOD)* of P and denote it by $SVOD(P)$. See Figure 2.6 for the SVOD of the polygon in Figure 2.5.

In the SVOD of a polygon, a bone due to r starts from r (called its *the origin*) on *DB* and extends to r_{cw} or r_{ccw} (called its *tips*) horizontally or vertically. Sometimes, a bone may go out of a boundary, such as *BB* and *RB*, and continue at the opposite boundary *TB* and *LB*, respectively. The intersections among barriers in the VOD become the intersections of the corresponding bones in the SVOD, as shown in the following.

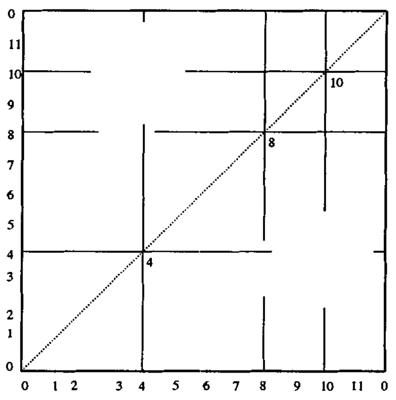


Figure 2.6: The SVOD of the polygon in Figure 2.5.

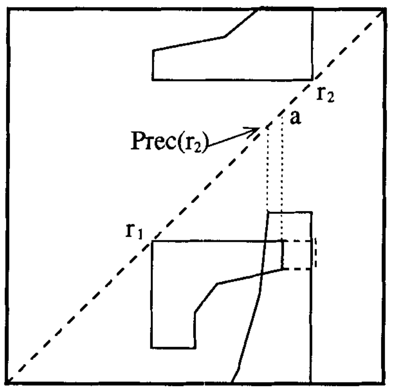


Figure 2.7: Two intersecting barriers.

One may think that two barriers intersect in the VOD but their corresponding bones may not intersect in the SVOD. We show such a situation in Figure 2.7. We have two barriers in the figure, with the southeast one due to r_1 and the northwest one due to r_2. For the sake of simplicity, we do not show other barriers. Suppose that $SE(r_1)$ intersects $NW(r_2)$, as shown in the figure. We can see that there is a point a such that a and r_1 are not mutually visible. However, if this is true, then r_1 cannot see any points between $Prec(r_2)$ and r_2. This means that the rightward extension of $SE(r_1)$ should extend further rightward, as shown by the dashed segments in the figure.

Another situation we consider is whether the curved sides of two barriers can intersect. We show this situation in Figure 2.8. In Figure 2.8 (a), $SE(r_1)$ intersects $NW(r_2)$ on their curved sides. We mark three points a, a' and a'' in the clockwise direction from a, as

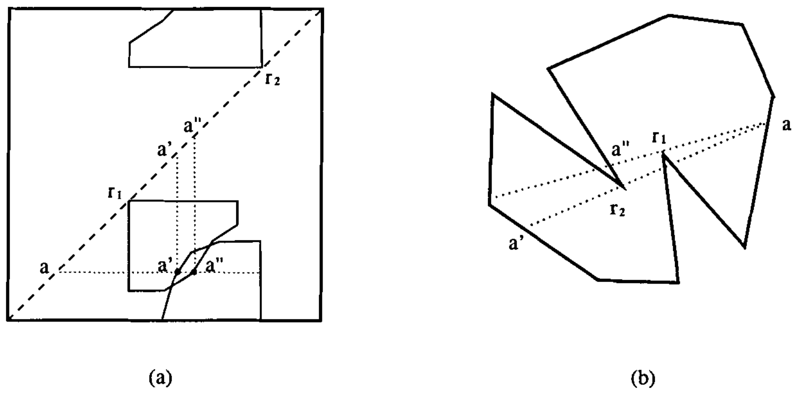


Figure 2.8: Two intersecting barriers.

shown in the figure. Now in order to make sure that the three points are in the correct order, we must have the situation shown in Figure 2.8 (b) in the polygon itself. Definitely, the situation shown is not possible since a'', the intersection point of the ray from a to r_1 and the boundary of the polygon, should be a point after r_2 in the clockwise direction, a contradiction to the fact that a'' precedes r_2 in the VOD. Thus, the curved sides of two barriers do not intersect.

Also it is easy to see that if two barriers do not intersect, the corresponding bones do not intersect either in the SVOD.

We will use VOD and SVOD to study and identify the "important" patterns that determine the searchability of a polygon. We will see how the VOD and SVOD of a polygon can be modified differently and employed to help us analyze different incarnations of polygon search problems.

2.5 Recontaminations

When we use different search models to search a polygon, we need to consider recontaminations. Recall from Chapter 1 that if one portion of a polygon was contaminated, became cleared and then becomes contaminated again, we say that the portion is recontaminated.

Let us consider recontaminations in the context of the VOD and SVOD of a polygon. We first consider the two-guard search model. Due to the requirement that the two guards be always mutually visible, the configuration encoding the current standing positions of the two guards must be a visible one, i.e., the corresponding point is always in the white area

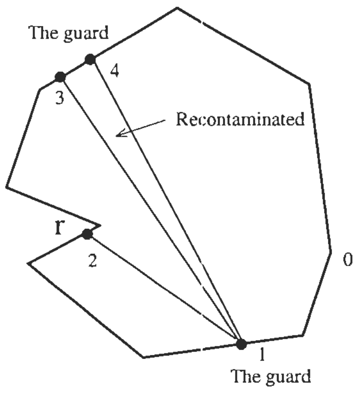


Figure 2.9: Recontamination in the two-guard search model.

of the VOD and SVOD. Some partial recontaminations are harmless, in the sense that the action is reversible. The situation is shown in Figure 2.9. Suppose that the two guards are at two points 1 and 4 on ∂P, respectively. The polygonal portion formed by $\partial P_{cw}(1,4)$ and the beam between the two guards is clear. When one guard moves from point 4 to point 3, some previously cleared portion of the polygon becomes recontaminated, as indicated in the figure. This recontamination is harmless. However, the guard cannot move from point 3 to point 2 while keeping the currently cleared portion clear (notice reflex vertex r in the figure), since this means that she loses her sight with the other guard at point 1 during the move, which is not allowed.

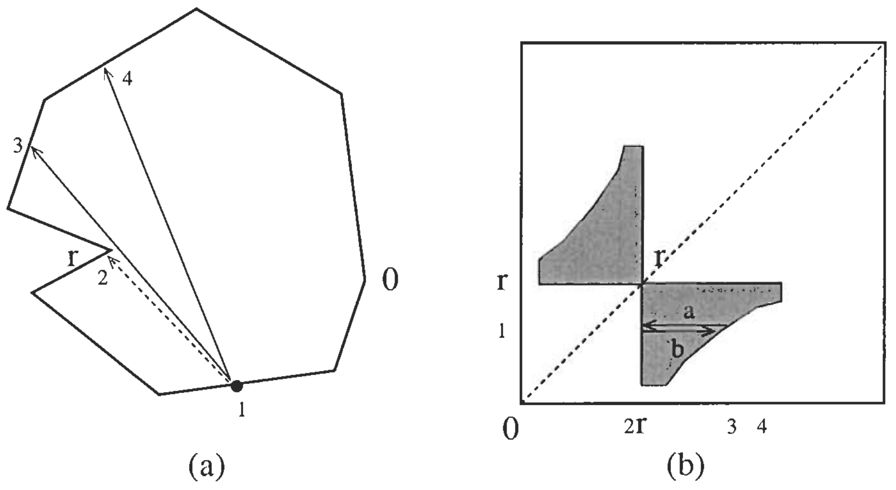


Figure 2.10: Recontamination in the 1-searcher search model.

As for the second search model by a boundary 1-searcher, the searcher and the beam head of her flashlight can backtrack. Consider the situation shown in Figure 2.10, where the searcher stands at point 1 while the beam head of her flashlight is at point 4. She rotates her flashlight to the left until the beam head is at point 3, at which time the beam grazes reflex vertex r. Thereafter, she continues rotating her flashlight until the beam head is at point 2, which is sufficiently close to r in the counter-clockwise direction. This is shown in Figure 2.10 (a). When we interpret this in the corresponding VOD (The y axis represents the current standing position of the searcher while the x axis represents the beam head of her flashlight.), it means that there is a jump over the downward extension of $SE(r)$ from right to left, as shown by the arrow marked as a in Figure 2.10 (b). We may also have a jump from left to right, as the segment marked as b in the figure. (However, this jump represents total recontamination, i.e., all the cleared area will be recontaminated, because the intruder could have been hiding in the triangular area formed by vertices r, 3 and the boundary of the polygon.) The reason that these jumps are considered is that while we require that the searcher be always on the boundary (this explains why we cannot have a jump over a barrier from top to bottom or vise versa), the beam head of her flashlight does not have to. We call such a jump a *beam head jump*. In the SVOD, such a beam head jump corresponds to the crossing of a vertical bone.

We will come back to the topic of recontamination in later chapters.

2.6 Previous work

The VOD and SVOD of a given P we discussed above are essentially the same as the VOD and SOD discussed in Section 1.3.5.

However, we observe that in the previous work these diagrams were mainly used to design a graph-based algorithmic solution to check whether a polygon is searchable by a 1-searcher. By studying these diagrams, we have realized that they provide us with more than the information for this purpose. As will be seen in later chapters, they are useful for a variety of search problems. Also the information hidden inside the diagrams helps us further explore the geometric properties of a given polygon, improve the performance of the previous solutions, and solve new interesting problems in the field.

Chapter 3

Room Search Problems

We consider room search problems in this chapter. We first study the room search problem by two boundary guards or searchers. (We use *guards* in the sequel to emphasize its evolution from the two-guard street search problem.) This is the first search model we discussed in Section 2.2. We then consider the room search problem under the second search model, i.e., a boundary 1-searcher. As will be seen, the latter problem is just an extension to the former. Under both search models, we study how to find all doors of a given polygon such that the resultant rooms are searchable.

Without causing any ambiguity, in this chapter, if we say that a room is searchable, it should be interpreted as that the room is searchable by two boundary guards.

3.1 Room search problem by two guards

Formally the two-guard room searchability problem is defined as follows.

Definition 3.1.1 *Suppose that d is the origin vertex of P. A room $P(d)$ is searchable by two boundary guards if there exist two continuous functions $l : [0,1] \to \partial P$ and $r : [0,1] \to \partial P$ such that (1) $l(0) = r(0) = d$ and $l(1) = r(1) \in \partial P$; (2) For any $x \in (0,1)$, $l(x) \prec_{cw} r(x)$ and $l(x)$ and $r(x)$ are mutually visible; and (3) At any time $x \in [0,1]$, $d \in \partial P_{cw}[r(x), l(x)]$. Deciding whether a room is searchable by two guards or not is called the room searchability problem.*

The functions $l(x)$ and $r(x)$ represent the current positions of the left guard and right guard on ∂P at $x \in [0,1]$, respectively. It is easy to see from the above definition that, if

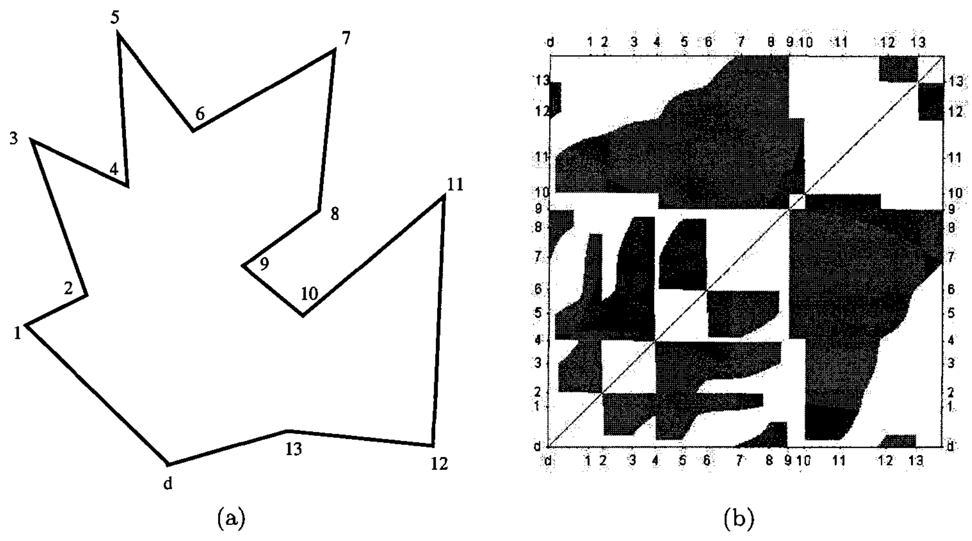


Figure 3.1: A room and its corresponding VOD.

a room is searchable, then for any time $x \in [0,1]$, the subpolygon formed by $\partial P_{cw}[r(x), d]$, $\partial P_{cw}[d, l(x)]$, and $\overline{l(x)r(x)}$ is clear. That is, it is guaranteed that no intruder can be in this subpolygon and thus the door is protected. On the other hand, if at sometime, the mutual visibility between $l(x)$ and $r(x)$ is violated, the intruder can then escape the room through d. Also Condition (3) in Definition 3.1.1 states that neither guards can go across the door.

3.2 The VOD and SVOD of a room

We will show that the VOD and SVOD introduced in Chapter 2 are helpful for solving the two-guard room searchability problem. As an example, Figure 3.1 shows a room with the door at d and its corresponding VOD.

Recall *DB*, *TB*, *RB*, *BB*, and *LB* defined for the VOD of a polygon in Section 2.3. For the room searchability problem, we drop the part below *DB* and only focus on the one above. For instance, for the VOD shown in Figure 3.1 (b), after this operation, we have a simplified diagram as shown in Figure 3.2. By abusing the notation a little, we call it the VOD of $P(d)$. As will be seen, this diagram (in a triangular shape) contains all the information we need to analyze the room searchability problem. We also label the relative positions of vertices along the diagonal line for easy reference.

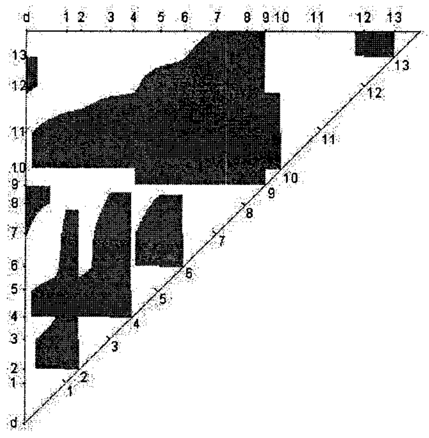


Figure 3.2: The VOD for the room shown in Figure 3.1.

We next consider the following conditions about the room search problem. (1) The left guard moves in the general clockwise direction while the right guard moves in the general counter-clockwise direction; (2) Both guards start from the door d; (3) Neither can go across the door; and (4) If the room is searchable, the two guards will meet at some point on ∂P. Let us look at how these four conditions are interpreted within the context of the VOD. According to the definition of the room search problem, at any time $x \in [0,1]$, we must have $l(x) \in \partial P$ and $r(x) \in \partial P$, and both $l(x)$ and $r(x)$ should be continuous. We consider the configurations $\langle l(x), r(x) \rangle$ as x changes from 0 to 1. Correspondingly, we use the notation (a,b) to represent a point inside the VOD.

The first three conditions can be represented in the VOD. Condition (1) tells us that the vertical axis of the VOD can be used to represent the current position of the right guard, while the horizontal axis represents the current position of the left guard. Condition (2) states that the positions of both guards are initially at (d,d), which is the top-left corner in the VOD. Condition (3) means that at no time can the positions of the two guards be outside the VOD. As for Condition (4), when the two guards eventually meet at some point at the end of the search, say g, on ∂P, then $|\partial P_{cw}[d,g]| + |\partial P_{ccw}[d,g]| = |\partial P|$. This means that the corresponding configuration is on DB of the VOD.

If a room is searchable, the sequence of configurations representing the positions of the two guards on ∂P can be mapped into the VOD as a continuous path through the white area in the diagram. Thus the VOD of a searchable room must contain a continuous path from the point (d, d) (the top-left corner) in the VOD to a point (p, q) on DB, and every point on this path is in the white area. Conversely, any continuous path, if there is any, through the white area in the VOD, which starts at the (d, d) and ends at DB, represents a continuous sequence of visible configurations, indicating that the room is searchable.

We call such a path a *legal path*. If no legal path exists in the VOD, the room is not searchable. We thus immediately have the following proposition.

Proposition 3.2.1 *A room is searchable if and only if there exists a legal path in its VOD.*

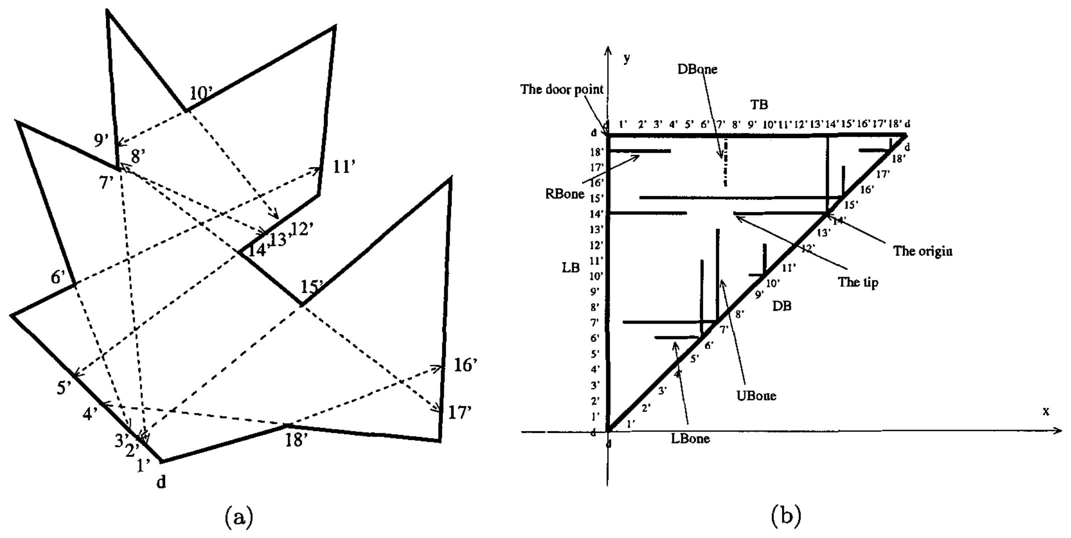


Figure 3.3: The room (shown in Figure 3.1 (a)) after relabeling some of its boundary points and the corresponding SVOD.

From the properties of the VOD of a polygon (a room in the current context) stated in Sections 2.3 and 2.4, we know that the SVOD of a room can be constructed from its VOD. In order to show the extensions representing the invisibility from each reflex vertex in the SVOD, we relabel the points on ∂P, taking only the reflex vertices into consideration and ignoring the non-reflex vertices. For instance, after relabeling, the room in Figure 3.1 and its corresponding SVOD are shown in Figure 3.3.

For the SVOD for a room, we use LB to represent its left side, TB for its top side, and DB for its diagonal, as before. We also call the bone segments, which are extending downward, upward, rightward, and leftward, the downward bones (*DBone*), the upward bones (*UBone*), the rightward bones (*RBone*), and leftward bones (*LBone*), respectively. Note that the downward and rightward bones here start at *TB* and *LB*, respectively. Recall the bones defined in Sections 2.3 and 2.4.

Also the special point (d, d) at the top-left corner of the SVOD is called the *door point*, to emphasize its correspondence to door d. We show these terms in Figure 3.3 (b). Note that the bone labeled as *DBone* in the figure does not exist in the original SVOD (that is why it is represented using a dashed line). We add it here only for an illustration purpose.

After this construction, the intersections of barriers in the VOD become the intersections of the corresponding bones due to the definitions of bones. (Recall the construction of the VOD of a polygon in Section 2.3.) The following proposition is immediate.

Proposition 3.2.2 *Given a room $P(d)$, there exists a legal path in its VOD if and only if there exists a path in its SVOD, which does not go across any bones, TB or LB, starting at the door point and ending at the same single point on DB.*

We still call such a path in the SVOD a *legal path*. Also for the sake of easy illustration, we treat *LB* and *TB* of the SVOD as the longest bones that a path should not go across. Due to this proposition, we can only focus on the SVOD in our following analysis.

3.3 The searchability of a room

The SVOD of a given room $P(d)$ represents the interactions of the bones corresponding to the invisible chains due to the reflex vertices of P. It is obvious that, if a region inside the SVOD is all surrounded by three or four bones (including *LB*, *TB*, and *DB*), a path can never enter the region if it is already outside. The two guards lose their mutual visibility when the path goes across a bone. For the same reason, if the path is already inside the region, it cannot go outside by crossing any surrounding bones. This is a key observation in the SVOD. We call such a region a *trap region*. We will identify them in the following.

For the sake of simplicity, in the following figures, if some vertices of P are irrelevant to our discussions, we will just omit them. We will draw a polygon as a circle and only mark the relevant reflex vertices on its boundary.

We list in Figure 3.4 all the possible patterns that make the door point inside a trap region. Note that the two arrows in Figure 3.4 (d) need not intersect. Any path within a

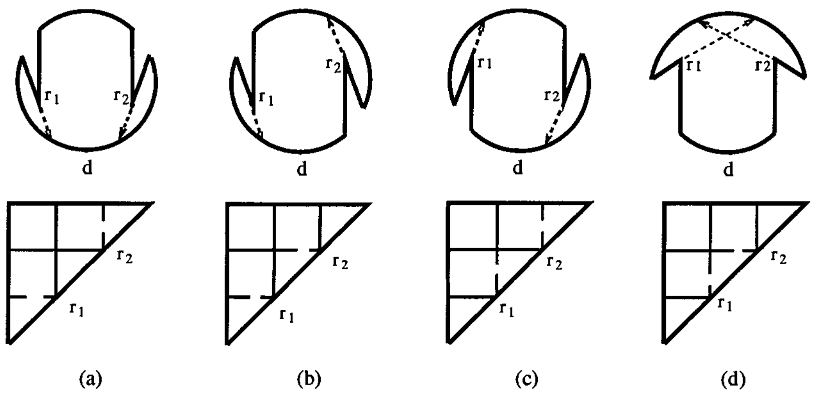


Figure 3.4: Four patterns that make the door point inside a trap region.

trap region can only be extended within it and cannot go outside. Note that these patterns also have effects on the accessibility of *DB*, as will be seen later on.

Lemma 3.3.1 *A room $P(d)$ is not searchable if the door point is inside a trap region formed by two bones, LB, and TB in its SVOD.*

Proof In the SVOD of a room, if the door point is inside a trap region formed by four bones (including *TB* and *LB* of the SVOD), any path starting at the door point cannot escape from the trap region. Otherwise, the path would go across one of the bones, which contradicts the requirement that the two guards be mutually visible or the requirement that the door be protected. ▮

If the door point is inside a trap region in the SVOD, we say that *d is trapped.*

Just as the door point can be trapped (inside a trap region), a point on *DB* can be inside a trap region (which is bordered by *DB*), too. The two guards cannot meet at this point without crossing bones and thus violating the mutual visibility requirement. We say that such a point is *unreachable*. We also say that the corresponding boundary point on ∂P is *unreachable.*

If every point on *DB* is unreachable, we then say that *DB* is unreachable. If a point on *DB* is not unreachable, then it is *reachable*. For example, any point between point $6'$ and point $7'$ on *DB* of the SVOD shown in Figure 3.3 is unreachable.

We list all the possible patterns in Figure 3.5 that generate a trap region bordered by *DB* of an SVOD. The following lemma follows from the above discussions.

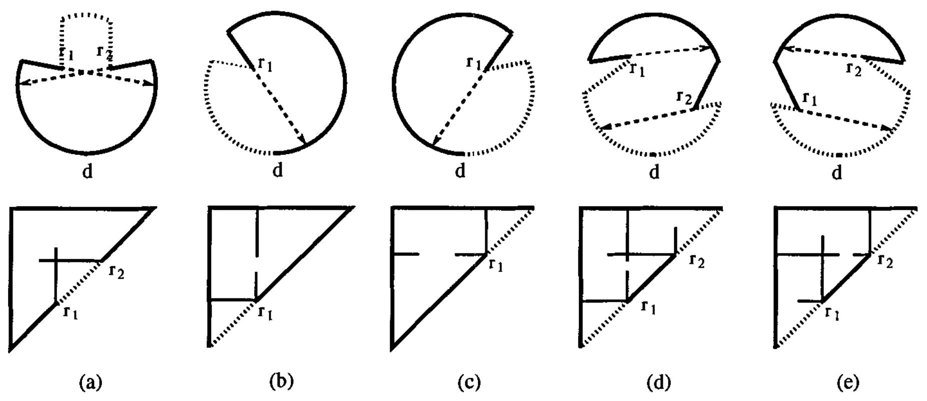


Figure 3.5: Five patterns that generate a trap region that is bordered by *DB*.

Lemma 3.3.2 *If every point on DB in the SVOD of a room P(d) is unreachable, then the room is not searchable by two guards.*

The negations of the conditions stated in Lemma 3.3.1 and Lemma 3.3.2, respectively, are also sufficient for a room to be searchable, as shown in the following lemma.

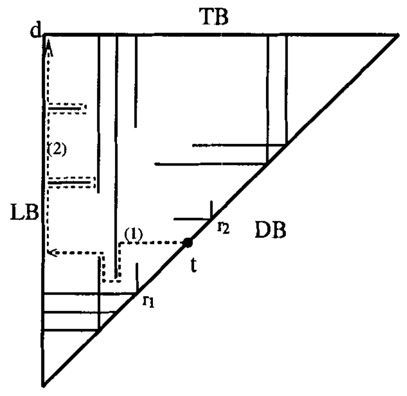


Figure 3.6: The sufficiency proof for room searchability.

Lemma 3.3.3 *For a room P(d), if neither the door point is trapped nor DB of its SVOD is unreachable, there must exist a legal path in the SVOD.*

Proof We will show in this proof that, given the conditions specified in the lemma, we can construct a path from t to the door point, where t is any point on *DB* of the SVOD that is reachable. We use Figure 3.6 as an illustration for our proof. Note that the SVOD in the figure is partial in the sense that we do not show all the bones and vertices. In the following discussions, we sometimes say that the path hits a bone. Actually, this hitting should be interpreted as coming *sufficiently close* to the bone.

Suppose that $r_1 \prec_{cw} t \prec_{cw} r_2$ on ∂P, where r_1 and r_2 are two reflex vertices and there is no reflex vertex between r_1 and r_2. We first extend the path horizontally toward *LB.*

This is the first step of our path construction. In this step, the path may encounter some vertical bones that prevent the path from going farther horizontally. Here we discuss the case where the path hits a downward bone. The case for an upward bone can be treated similarly. We know that this bone cannot intersect the *LBone* due to r_1, since otherwise t is unreachable in the SVOD. (The downward bone and *LBone* due to r_1 form a trap region, as shown in the pattern in Figure 3.5 (d), bordered by *DB.* t is inside the region.) For the same reason, it cannot intersect any leftward bone due to r', where $r' \prec_{cw} r_1$. Furthermore, any rightward bones intersecting this bone would make the door point trapped. We are thus able to extend the path downward. When the path reaches this downward bone's tip, it then resumes going horizontally until it hits the next vertical bone. Using this argument repeatedly, the path is able to reach *LB* of the SVOD eventually. This step is marked as (1) in the figure.

We then switch to the next step (marked as (2) in the figure). At the beginning of this step, the path is at some point sufficiently close to *LB.* The path now goes upward along *LB.* In this process, the path might encounter horizontal bones. But we first claim that those horizontal bones should not be leftward ones. There are two cases we have to consider here. If it is a leftward bone due to a reflex vertex r', where $r' \prec_{cw} t$, the bone should touch both *DB* and *LB* of the SVOD. Since the path is being extended upward, we know that the path should be above the leftward bone sometime before. This results in a contradiction, since it means that the path has gone across the bone before. If the leftward bone is due to r'', where $t \prec_{cw} r''$, then the fact that this leftward bone should touch both *DB* and *LB* means that t is unreachable, a contradiction again.

For any rightward bone, the path can then follow it rightward horizontally. Using the same reasoning as above, we know that this rightward bone cannot hit any upward bone, since, if so, the path was already inside the trap region generated by the rightward bone, the upward bone, *LB* and *DB*, a contradiction to the fact that the path was constructed without crossing any bones. The path thus can reach the tip of the rightward bone. By going around the tip, the path follows the bone leftward.

Now we consider the downward bones. But any intersection between some downward bone and this rightward bone means that the door point is trapped. So the path will eventually be able to reach *LB* of the SVOD again. Repeating this process, the path will finally reach the door point.

By reversing the path we just constructed, we have a legal path and the lemma follows. ▮

Summarizing the above discussions, we have the following result.

Theorem 3.3.4 *A room $P(d)$ is searchable by two guards if and only if neither the door point is trapped nor DB of its SVOD is unreachable.*

3.4 Checking the searchability

We begin with a lemma which shows the possible range for the final meeting point of the two guards with respect to the relationship between the door d and a reflex vertex r.

Lemma 3.4.1 *Let r be a reflex vertex of a room $P(d)$. If $d \notin P_{cw}(r)$ ($d \notin P_{ccw}(r)$, resp.), the final meeting of the two guards can only be in the polygonal chain $\partial P_{ccw}(d, r)$ ($\partial P_{cw}(d, r)$, resp.).*

Proof Refer to [89, 93] for one possible proof. It can also be proved using the SVOD with the obvious observation that the bone due to r crosses from *DB* to *LB* horizontally (*TB* vertically, resp.). See Figures 3.5 (b) and (c), respectively. ▮

Recall from Section 1.3.11 that if P is *LR*-visible, we can find a set of pairs (A_i, B_i) such that for any $s \in A_i$ and $t \in B_i$, P is *LR*-visible with respect to (s, t). Given a room $P(d)$, we now discuss the relationship between d and these (A_i, B_i) pairs.

The following lemma shows the relationship between the searchability of a room and its *LR*-visibility.

Lemma 3.4.2 *If a room $P(d)$ is searchable by two guards, then P is LR-visible.*

Proof Let $t \in \partial P$ be the final meeting point. Suppose that P is not *LR*-visible. Then $\partial P_{cw}(d, t)$ and $\partial P_{ccw}(d, t)$ are not mutually weakly visible. From a result in [36], there exists a component which contains neither d nor t.

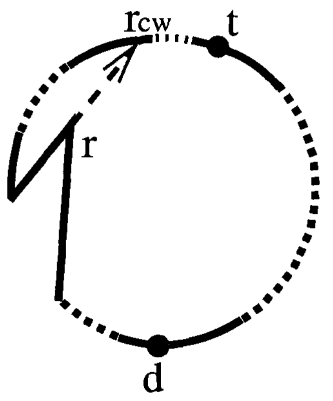


Figure 3.7: The illustration of the proof for Lemma 3.4.2.

Without losing generality, suppose that it is a clockwise component due to a reflex vertex r (If there are several such components, we select the one whose originating reflex vertex is the closest to t in the counter-clockwise direction.). This is shown in Figure 3.7. Since $P(d)$ is searchable and the two guards cannot lose their mutual visibility, so they must be both in $P_{cw}(r)$ simultaneously at some time point, from which we also know that the polygonal chains $\partial P_{cw}(d, r)$ and $\partial P_{ccw}(d, r_{cw})$ are mutually weakly visible.

We select a point, denoted by t', on $\partial P_{cw}(r, r_{cw})$ which is sufficiently close to r_{cw} in the counter-clockwise direction. We claim that if $P(d)$ is searchable, then P is LR-visible with respect to d and t'.

In order to show this, we only need to show that any point on $\partial P_{cw}(r, t')$ is visible to some point on $\partial P_{ccw}(d, t')$. Suppose that there is a point on $\partial P_{cw}(r, t')$ that is not visible to $\partial P_{ccw}(d, t')$. Then we only have the following two situations. (1) The point is hidden by a counter-clockwise component due to a reflex vertex r' ($r' \in \partial P_{cw}(r, t')$) whose r'_{ccw} is on $\partial P_{cw}(d, r')$. However, then by Lemma 3.4.1, the final meeting point can only be on $\partial P_{cw}(d, r')$, which is a contradiction to the fact that the final meeting point is at t. (2) The point is hidden by a clockwise component due to a reflex vertex r'' ($r'' \in \partial P_{cw}(r, t')$) whose r''_{cw} is on $\partial P_{cw}(r'', t')$, which means that $P_{cw}(r'')$ does not contain t. However, this again leads to a contradiction to the fact that $P_{cw}(r)$ is the "closest component" (we selected it as above) that does not contain t.

We thus have shown that $\partial P_{cw}(r, t')$ is weakly visible to $\partial P_{ccw}(d, t')$. Due to the selection of t', all these mean that $\partial P_{cw}(d, r_{cw})$ is weakly visible to $\partial P_{ccw}(d, r_{cw})$. We thus conclude that P is LR-visible with respect to d and r_{cw}. ▮

Corollary 3.4.3 *For a room $P(d)$, if P is not LR-visible, then $P(d)$ is not searchable by two guards.*

Lemma 3.4.4 *If a room $P(d)$ is searchable, then d must be on some A_i or B_i.*

Proof From Lemma 3.4.2, we know that if $P(d)$ is searchable, then there must exist a point $t \in \partial P$ such that P is LR-visible with respect to d and t. So d must be on some A_i or B_i. ∎

Given a room $P(d)$, we first check its LR-visibility. If it is not LR-visible, we know that it is not searchable by two guards. If it is LR-visible, then we check whether door d is in any A_i or B_i and decide whether the room $P(d)$ is searchable accordingly. Clearly this checking can be done in $O(n)$ time by visiting A_i and B_i once (since A_i and B_i pairs can be precomputed in linear time, as shown in Section 1.3.11).

In the following discussion, we always assume that d is on some A_i or B_i. We then relabel the A_i's and B_i's such that the pair containing d is indexed by 0.

We first consider the four possible patterns in Figure 3.4. By the assumption that d is on A_0, we immediately know that the patterns in Figure 3.4 (a), (b) and (c) are not possible, since they imply that the corresponding B_0 must be simultaneously present in two disjoint components.

The pattern in Figure 3.4 (d) is exactly an s-deadlock (d is treated as s), which, as shown in Section 1.3.3, can be precomputed in linear time. Clearly, given a room $P(d)$, to see whether d is in an s-deadlock can be checked in $O(n)$ time by traversing ∂P once.

We then consider the patterns that generate trap regions bordered by DB, as shown in Figure 3.5. We treat each pattern as follows.

The pattern in Figure 3.5 (a) involves a deadlock. Finding all of the deadlocks on ∂P (they are unreachable) can be done in linear time, as shown in Section 1.3.3.

As for the pattern in Figure 3.5 (b), we will, starting from d in the clockwise direction, find the farthest clockwise component (due to a reflex vertex r) which does not contain d. With the shortest path tree precomputed from d, this can be done in linear time by traversing the boundary of P and checking each clockwise component. The polygon chain $\partial P_{cw}(d, r)$ is marked as unreachable.

The pattern in Figure 3.5 (c) is symmetric to the pattern in Figure 3.5 (b).

As for the pattern in Figure 3.5 (d), we notice that the corresponding B_0 should lie inside the clockwise component due to r_1. The relationship between r_1 and r_2 can be described as follows. Let r_{1cw} and r_{2cw} be the backward ray end point from r_1 and r_2, respectively. We look for a pair of reflex vertices r_1 and r_2 such that (a) $d \in P_{cw}(r_2)$ and $d \notin P_{cw}(r_1)$; (b) $r_{2cw} \prec_{cw} r_1$; and (c) $r_{1cw} \prec_{cw} r_2$. We want to find, from d, the farthest clockwise component (due to r_1) in the clockwise direction from d, and the farthest clockwise component (due to r_2) in the counter-clockwise direction from d that satisfy the above three conditions.

We preprocess the polygon with respect to A_0 and B_0 by computing the shortest path tree from d to any other vertices and the shortest path tree from a point $t \in B_0$ to any other vertices. These can be done in linear time. See Section 1.3.11 for more information about this computation. Note that now P is LR-visible with respect to d and t.

We first consider the clockwise components in the clockwise direction from d to t. We immediately know that we only need to consider the non-redundant ones since if there is any redundant component that satisfies the above three conditions, we can always find a non-redundant one that is even farther. We move along ∂P in the clockwise direction from d and check the non-redundant components. We only consider those that do not contain d, which can be checked in $O(1)$ time. These components have the property that if the clockwise components due to r' and r'' are two such components and $r' \prec_{cw} r''$, then $r'_{cw} \prec_{cw} r''_{cw}$, and for any r whose clockwise component is such a component, $r_{cw} \in \partial P_{ccw}(d, t)$. For these components, we denote their originating reflex vertices in a set called L_{cw}. We maintain a pointer P_l for traversing them. P_l starts at the first vertex in L_{cw} and moves from d to t in the clockwise direction.

We next consider the clockwise components in the clockwise direction from t to d. It is obvious that this time we have to consider all the clockwise components since each of them could be possibly the farthest. We check each component to see whether d is inside. This can be done in $O(1)$ time, after the shortest path tree from d is precomputed in linear time. We exclude those that do not contain d from further consideration. We also check each selected component to see whether t is contained inside. We keep those that do not contain t. Thus all the remaining components contain d but not t. For these components, we put their originating reflex vertices in a set called R_{cw}. We also maintain a pointer P_r for moving over them. P_r starts at $P_{l_{cw}}$ (since we do not need to consider the components whose originating reflex vertices are inside the clockwise component due to P_l), and moves in the clockwise direction from t to d.

Refer to Figure 3.8, which shows P_l and P_r and their move directions. The algorithm is described as follows.

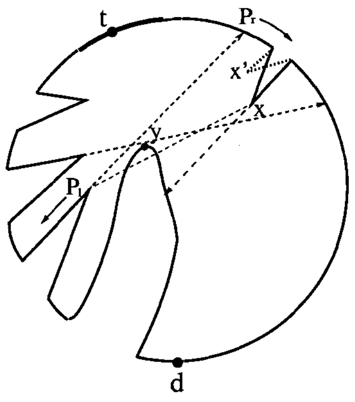


Figure 3.8: Illustration for the algorithm for detecting the pattern in Figure 3.5 (d).

P_r moves to the next reflex vertex x in R_{cw}. Note that in this process, if P_r goes across any backward ray end point from a reflex vertex in L_{cw}, we update P_l to it.

We know that the backward ray end point from x can intersect neither $\partial P_{cw}(t, x)$ nor $\partial P_{cw}(x, d)$. We only need to test whether the backward ray intersects $\partial P_{cw}(d, P_l)$ or $\partial P_{cw}(P_l, t)$. If the latter is true, we give up x and continue to the next element in R_{cw}. If the former is true, we then will move P_l to the next element in L_{cw}.

In order to do so, we first check whether P_l and x are mutually visible. By preprocessing (See [56] for details for how to preprocess a street in order to check the mutual visibility between two points on the two opposite boundaries in $O(1)$ time.) the polygon with respect to (d, t) in $O(n)$ time [56], this checking can be done in $O(1)$. If they are mutually visible, then we only need to check whether the ray is to the right or left of the segment $\overline{xP_l}$. If the former is true, it means that the backward ray intersects $\partial P_{cw}(P_l, t)$. Otherwise, the ray intersects $\partial P_{cw}(d, P_l)$.

Suppose that P_l and x are not mutually visible. In order to check whether the backward ray from x intersects $\partial P_{cw}(P_l, t)$ or $\partial P_{cw}(d, P_l)$ in constant time, we maintain another pointer P_x which starts from P_l and moves in the counter-clockwise direction from P_l toward d. P_x goes through each vertex and checks its visibility with x in constant time. We know that eventually P_x will point to a reflex vertex y, since otherwise d is not contained in the clockwise component from x. Clearly y is a reflex vertex. We associate y with P_l. We then check the backward ray from x with the segment $\overline{xy}$, from which we can easily check to see whether the ray intersects $\partial P_{cw}(P_l, t)$ or $\partial P_{cw}(d, P_l)$, as discussed above.

We then consider the following two situations. (1) Suppose that we have found an x such that the conditions for the pattern in Figure 3.5 (d) are satisfied. We save P_l. We next move P_l to the next reflex vertex to see whether the component from this new P_l contains x or not, since we desire to find the farthest one. If x is contained in the component, then we do not need to go farther since any farther components must also contain x. The saved P_l and x are the pair we are looking for. We mark the polygonal chain $\partial P_{cw}(x, d)$ and $\partial P_{cw}(d, P_l)$ as unreachable. If x is not contained in the component, we save the current P_l and move P_l to the next one. We then continue the process. (2) Suppose that no such x has been found and P_r goes across a backward ray hit point due to a reflex vertex in L_{cw}. We update P_l to the next reflex vertex in L_{cw}, and P_r to be the hit point, and whole process continues.

One important fact here is when we repeat the above process with new P_l and P_r, we might need to move the pointer P_x again in the counter-clockwise direction. However, any previously checked polygonal chains will not need to be checked again. For instance, the polygonal chain $\partial P_{ccw}(P_l, y)$ in Figure 3.8 will not be checked again for any new P_l and P_r since we already knew that any vertices on $\partial P_{ccw}(P_l, y)$ would not be visible to any point on $\partial P_{cw}(x, d)$. Thus any vertex on $\partial P_{cw}(d, t)$ can be checked at most once. Also, P_r is always advanced in the clockwise direction on $P_{cw}(t, d)$. Thus we conclude that we can detect the pattern in Figure 3.5 (d) in $O(n)$ time.

The pattern in Figure 3.5 (e) is symmetric to the pattern in Figure 3.5 (d).

After the unreachable parts on the boundary of P have been marked, checking whether there is any part that is not marked can be done in linear time by traversing the boundary once. Thus, we have the following theorem.

Theorem 3.4.5 *For a room $P(d)$ with n vertices, we can check whether it is searchable by two guards in $O(n)$ time.*

3.5 Finding all doors associated with a polygon

A natural followup to the room searchability problem, the *Find-All-Doors problem* is to find all the doors for a given polygon P such that the resultant rooms are searchable.

Now that we know from Section 3.4 that the searchability of a room $P(d)$ can be checked in $O(n)$ time, a straightforward solution to the Find-all-Doors problem for a polygon can be found in $O(n^2)$ by checking each vertex in turn to see whether it can be the door. However, as will be seen, it can be done much faster.

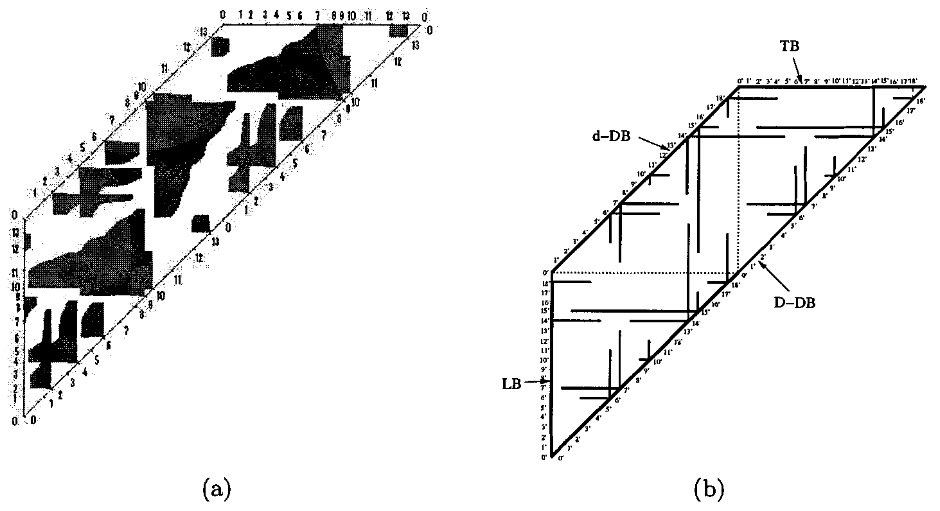


Figure 3.9: The CVOD and CSVOD of the polygon shown in Figure 3.2

As shown in Section 3.3, for a given room, we focus on the above-diagonal half of its VOD and SVOD to conduct our analysis. As has been shown, it is enough for checking whether the room is searchable or not. But in order to study the Find-All-Doors problem, including the below-diagonal half facilitates our analysis greatly, as described in the following construction.

For a given simple polygon P, with vertices p_0, p_1, $\cdots$, p_{n-1} in the clockwise direction starting at p_0, we first construct its VOD and SVOD starting from p_0. Recall Sections 2.3 and 2.4. We move the bottom-right triangle below *DB* of the VOD to the left side of the top-left triangle, keeping the two triangles abreast. We then copy the top-left triangle underneath the bottom-right triangle. We construct the corresponding SVOD similarly.

We call the diagram thus constructed from the VOD the *complete visibility obstruction diagram* (CVOD) and the skeletal diagram from the SVOD the *complete skeletal obstruction visibility diagram* (CSVOD) of P. For example, for the polygon shown in Figure 3.1, the corresponding diagrams are shown in Figure 3.9.

Again we focus our attention on the CSVOD of P for the analysis of the Find-All-Doors problem. There are four sides of a CSVOD. By abusing the notation a little, we still call the top side the *top boundary*, denoted as *TB*, and the left side the *left boundary*, denoted as *LB*. There are two diagonal sides in the CSVOD. We call the shorter diagonal side the *door boundary* and denote it as *d-DB*, and the longer diagonal side the *destination boundary* and denote it as *D-DB*. It is easy to see that, by following ∂P in the clockwise direction starting at p_0, *d-DB* represents one traversal of ∂P while *D-DB* represents two traversals of ∂P from p_0. As will be seen, the CSVOD of P encodes sufficient information for analyzing the Find-All-Doors problem.

We first analyze the patterns that generate trap regions bordered by *D-DB*, thus making some parts on it *unreachable*. We then analyze the patterns that generate trap regions bordered by *d-DB* such that the points on *d-DB* within these regions are *trapped* and cannot be used as a door. Here the definition of a trap region is similar to the one defined in Section 3.3, i.e., a region inside CSVOD being surrounded by bones (including *d-DB*, *D-DB*, *TB* and *LB*).

We have to emphasize that P must be *LR*-visible, otherwise there is no searchable room that we can find in it.

3.5.1 Patterns bordering on *D-DB*

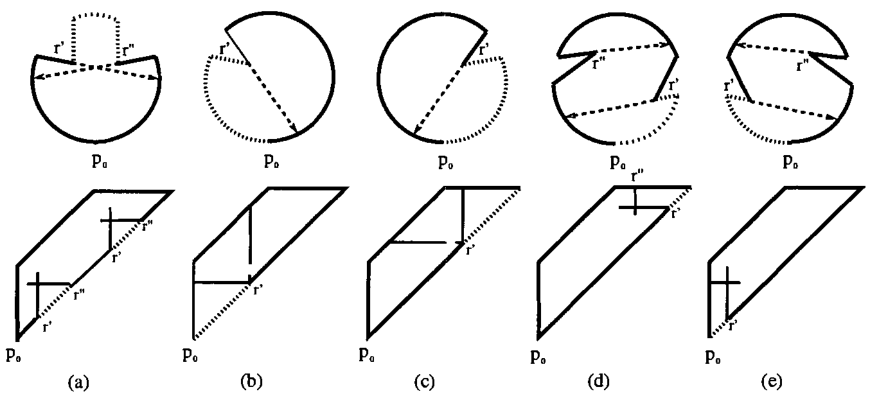


Figure 3.10: The patterns that generate trap regions bordered by *D-DB*.

We list in Figure 3.10 all the possible patterns that make some parts on *D-DB* unreachable. Note that we just draw their effects on *D-DB*. They might also cause trap regions on *d-DB*, which shall be analyzed below. It is obvious that those patterns are similar to the ones in Figure 3.5, and we can compute all of them in $O(n)$ time and mark the corresponding parts on *D-DB* as unreachable. In order to facilitate our later computations, we maintain a list called D_L for all the reachable parts on *D-DB*. Each element in this list is identified by a pair of starting and ending points. Later on we will use D_L.

3.5.2 Patterns bordering on *d-DB*

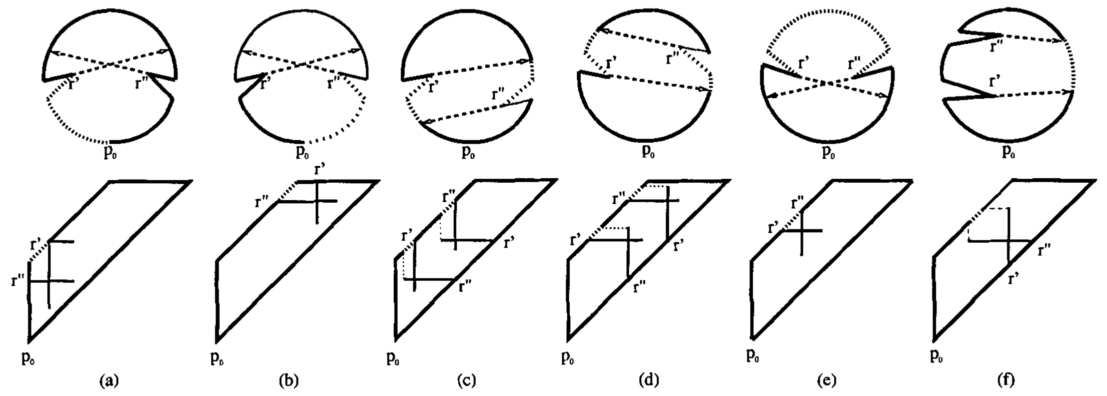


Figure 3.11: The patterns that generate trap regions bordered by *d-DB*.

Correspondingly, all the possible patterns that can generate trap regions bordered by *d-DB* (thus making the points inside trapped) in the CSVOD of P are shown in Figure 3.11. Also note that, by the above discussions, any potential doors for a given polygon can only be on A_is' and B_is'. We thus do not consider any other parts on ∂P.

The patterns in Figure 3.11 (a) and (b) correspond to deadlocks, which can be computed in $O(n)$. Pattern (c) can be ignored since we know that any door should be on A_i or B_i, and if there is any A_i on $\partial P_{cw}(r''_{cw}, r')$, the corresponding B_i cannot be in the two disjoint clockwise components due to r' and r'' simultaneously. Pattern (d) can be dealt with using a similar reasoning. The pattern in Figure 3.11 (e) corresponds to a deadlock, which can be identified in $O(n)$. Pattern (f) is not possible for the same reason as the one for patterns (c) and (d).

Note that some A_i and B_i can intersect the parts just computed as above. We need not consider the vertices in A_i or B_i that are in these computed parts. We put all the remaining vertices in P_d (the potential doors). In order to find all the doors such that the resultant rooms are searchable, we only need to focus on the vertices in P_d.

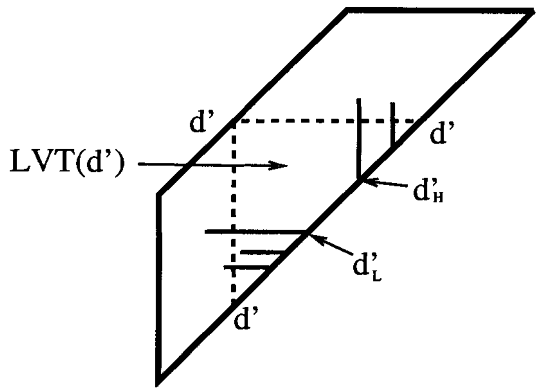


Figure 3.12: The local visibility triangle of a vertex d'.

3.5.3 Local visibility triangle

Intuitively speaking, any vertex d' in P_d represents a potential door, and in order to check the searchability of $P(d')$, we need to consider, in the CSVOD, the triangle formed by d' and its two counterparts on *D-DB*. We call such a triangle the *local visibility triangle* of d', which extends horizontally and vertically from *d-DB* to *D-DB*. We denote it as *LVT*(d'), as shown in Figure 3.12.

For each vertex $d' \in P_d$, we have to consider two situations. First, we need to compute from d' to d' in the clockwise (counter-clockwise, resp.) direction the farthest reflex vertex d'_L (d'_H, resp.) whose clockwise component (counter-clockwise component, resp.) does not contain d'. Note that we only need to focus on the non-redundant components, which can be identified in linear time. By traversing ∂P in the clockwise direction twice and in the counter-clockwise direction twice, respectively, those two farthest reflex vertices can be obtained collectively for all the vertices in P_d in $O(n)$. Note that if such a farthest component does not exist, we set d'_L or d'_H to be d' itself. If $d'_H \prec_{cw} d'_L$ starting from p_0, we immediately know that d' cannot be a door.

Second, for each $d' \in P_d$, we need to check whether the patterns shown in Figure 3.5 (d) and (e) exist in *LVT*(d'). Here we only discuss the situation for pattern (d). The situation for pattern (e) is treated similarly. We use Figure 3.5 (d) as our reference in the following proof.

Lemma 3.5.1 *For any selected $d' \in P_d$, if pattern (d) exists for d', then all the other vertices in P_d are either in $P_{cw}(r_1)$ or in $P_{cw}(r_2)$.*

Proof Suppose that a $d'' \in P_d$ is neither on $P_{cw}(r_1)$ nor on $P_{cw}(r_2)$. From the computation of P_d above, there must exist an A_i that contains d''. We immediately know that the corresponding B_i should be in $P_{cw}(r_1)$ and $P_{cw}(r_2)$ simultaneously, which is not possible. This is because the pair (A_i, B_i) is the output from the algorithm we discussed in Section 1.3.11. Thus any $d'' \in P_d$ should be either on $P_{cw}(r_1)$ or on $P_{cw}(r_2)$. ▌

With this lemma, in order to detect pattern (d) we can select any $d' \in P_d$ and use the algorithm in Section 3.4 to detect r_1 and r_2.

If no such pattern exists for d', there is no such pattern for all vertices in P_d. Otherwise, for any d'', where $d'' \in P_d$ and $d'' \in \partial P_{cw}(r_2, r_{2cw})$, we need to check whether $r_2 \prec_{cw} d''_H$. If true, we need to replace d''_H with r_2. We leave d''_L unchanged since it is already the farthest. Also for any d''', where $d''' \in P_d$ and $d''' \in \partial P_{cw}(r_1, r_{1cw})$, we repeat the above process to see whether d'''_H also needs to be changed.

Clearly after the computation of r_1 and r_2, which should be done once, for each $d' \in P_d$, we can compute the possible meeting place for d', which is $\partial P_{cw}(d'_L, d'_H)$. We do this for every $d' \in P_d$, which takes $O(n)$ time.

Again, after this computation, if $d'_H \prec_{cw} d'_L$ starting from p_0, we immediately know that d' cannot be a door. Note that even though we have $d'_L \prec_{cw} d'_H$ for some d', some part on the polygonal chain $\partial P_{cw}(d'_L, d'_H)$ might not be in D_L. So at this moment, we are still not sure whether d' can be used as a door for a searchable room.

3.5.4 Finding all rooms

With the sets P_d and D_L ready, we now show how to find all doors for searchable rooms in a given polygon in linear time.

We group the vertices in P_d according to their d'_L, in the clockwise direction starting at p_0. We maintain two pointers in the following algorithm. *PR* traverses the elements in D_L, while *PD* goes through each group in P_d.

Algorithm 3.5.2

```
1. PR is set to the first element in $D_L$;
2. PD is set to the first group in $P_d$; /* We call each potential door $d'$ below. */
3. Loop:
4.     if PR.EndPoint $\prec_{cw}$ PD.$d'_L$ then
```

```
5.      $PR = PR.next$;
6.      if $PR$ is null then
7.          Output all the remaining vertices in the
8.          groups starting at $PD$ as not doors;
9.      else
10.         goto Loop;
11.  else
12.     if $PR.StartPoint \prec_{cw} PD.d'_L$
13.         $x = PD.d'_L$;
14.     else
15.         $x = PR.StartPoint$;
16.     for each element $d''$ in the group pointed by $PD$
17.         if $x \prec_{cw} d''_H$ then
18.             output $d''$ is a door;
19.         else
20.             output $d''$ is not a door;
21.     end for
22.     $PD = PD.next$;
23.     if $PD$ is null then
24.         stop;
25.     else
26.         goto Loop;
```

Note that pointer *PD* points to a group of vertices in P_d that have common d'_L. Lines (4) to (10) try to find the first reachable part on ∂P that covers d'_L associated with the current group of potential doors. Lines (12) to (20) check, for each individual vertex in the current group, whether this reachable part also covers the corresponding d'_H. If true, then by Theorem 3.3.4 the room with door d'' is searchable. Otherwise it is not searchable. The reason behind this claim is that *PR* points to the first reachable part that might intersect $\partial P(d''_L, d''_H)$. If its *StartPoint* is preceded by d''_H, then d'' cannot be used as a door for a searchable room and we need to check no other elements in D_L. Lines (21) to (25) look for the next group of potential doors and loop again.

PR will traverse forward through D_L once, which is $O(n)$. Also pointer PD always moves forward, in the clockwise direction from p_0, and traverses through P_d once, which takes time $O(n)$. Thus we have the following theorem.

Theorem 3.5.3 *For a given polygon P with n vertices, it takes $O(n)$ time to solve the Find-All-Doors problem, where n is the number of vertices of P.*

3.6 Discussions

Our result regarding the searchability of a room was also reported in [89]. However, our approach is more concise. Furthermore, using the VOD and SVOD of a room and with the help from some previous work we have improved the searchability checking time to $O(n)$. We have also investigated a new problem for finding all doors in a polygon. As discussed above, the new problem can also be solved in linear time.

Another advantage of our approach is that we can easily extend it to the room search problem by a 1-searcher (See Section 2.2 for the two search models.), as shown below.

3.6.1 Extension to the room search problem by one 1-searcher

With a 1-searcher, we consider that she moves from d along ∂P in the general counter-clockwise direction and d must be between her and the beam head of her flashlight in the clockwise direction. Again, at anytime the door d must be protected.

The power of a 1-searcher is stronger than that of two guards, which is quite counter-intuitive, since there is one more searcher in the two-guard search model. The reason behind this is that in the search problem by a 1-searcher, the cleared area in the room can be recontaminated because of the recontamination discussed in Section 2.5. Crossing a barrier from right to left is allowed (marked by a in Figure 2.10 (b)). However, this recontamination is not allowed in the model where we have two guards since the guards are required to always stay on the boundary.

It is also obvious that the conditions specified in Theorem 3.3.4 are all applicable to the room search problem by a 1-searcher, except for the pattern in Figure 3.5 (e) (due to the recontamination as discussed above), where the polygonal chain $\partial P_{cw}(p_0, r_1)$ can be a possible place that a search ends.

We thus conclude that all the discussions about the room search problem by two guards can be applied to the problem by a 1-searcher. We have the following theorem.

Theorem 3.6.1 *A room $P(d)$ is searchable by a 1-searcher if and only if neither d is trapped nor DB is unreachable in the SVOD. The searchability can be checked in $O(n)$ time. Furthermore, it takes $O(n)$ time to find all the rooms associated with a polygon that are searchable by a 1-searcher.*

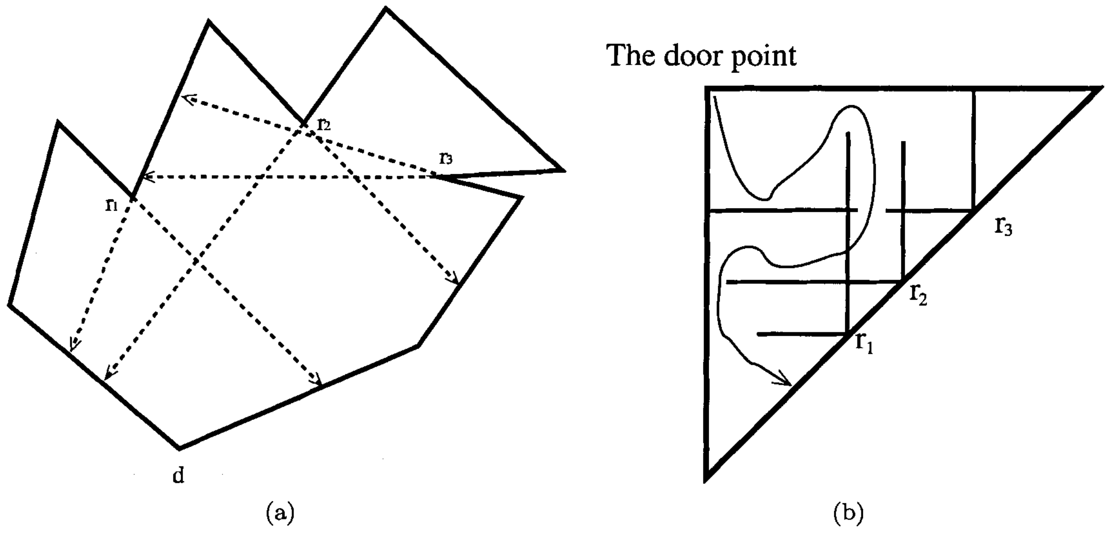


Figure 3.13: A room searchable by a 1-searcher but not searchable by two guards.

In Figure 3.13, we depict a room and its corresponding SVOD, which is searchable by a 1-searcher but not by two guards. The legal path can be constructed by crossing the upward bone due to r_1. However, as discussed above, such a crossover is not allowed when the room is searched by two guards. For the sake of simplicity, we just show the relevant vertices in the figure.

The room search problem by a 1-searcher was also investigated in [74]. But again, a comparison between their work and ours here indicates that our approach is much simpler, an easy extension of the situation for the room search problem by two guards.

Chapter 4

Polygon Search by a 1-Searcher

Given a simple polygon P, suppose that we have a boundary 1-searcher, who always stays on the boundary of the polygon. Her task is to search the polygon using her flashlight in order to catch an intruder roaming in the polygon. See Section 2.2 for the introduction to this search model.

If, no matter where the intruder starts and how he moves, he is eventually detected by the 1-searcher, we say that the polygon is *1-searchable*. In this chapter, if we say that a polygon is searchable, it should be interpreted as that the polygon is searchable by a boundary 1-searcher. Also a searcher should be understood as a boundary 1-searcher.

4.1 Introduction

The searcher and the intruder are modeled as points that can move continuously. Let $e(t) \in P$ denote the position of the intruder at time $t \geq 0$. It is assumed that $e : [0, \infty) \rightarrow P$ is a continuous function, representing the unpredictable move path of the intruder, who is capable of moving arbitrarily faster than the searcher (in order to make the problem non-trivial). The initial position $e(0)$ and the move path e are unknown to the searcher.

Let γ represent a continuous path of the searcher in the form $\gamma : [0, \infty) \rightarrow P$ and $\gamma(t)$ denote the position of the searcher at time $t \geq 0$. The searcher's visibility is restricted to the beam emanating from her flashlight at $\gamma(t)$, whose direction can be changed continuously with a bounded angular rotational speed. Let $\theta(t)$ denote the beam head of the flashlight at time t on ∂P. The beam head is the point where the beam from the searcher's flashlight intersects the polygon boundary before it leaves the polygon for the first time.

The polygon P is *1-searchable* if we can always find a *search schedule* (γ,θ) satisfying the condition that for every continuous function $e : [0, \infty) \rightarrow P$, there exists a time $t \in [0, \infty)$ such that $e(t) \in \overline{\gamma(t)\theta(t)}$.

Note that there is a major difference between the 1-searcher polygon search problem in this chapter and the two-guard room search problem in Chapter 3. In the two-guard room search problem, at any time the door should be protected, which means that the polygonal portion containing the door should be always clear. However, in the 1-searcher polygon search problem, we do not have such a restriction.

Some previous work has investigated the problem, as stated in Chapter 1. One previous approach was to identify some forbidden patterns inside a polygon, whose presence renders the polygon non-searchable. For some work using this approach, see [91, 92, 106, 109]. Usually this type of approach involves a case-by-case analysis which is quite lengthy and complex. Another line of approach is to use the visibility obstruction diagram (VOD) (See Section 1.3.5 for detailed discussions.) to help solve the problem. But the approach was algorithmic in the sense that it circumvented the involved case-based analysis and only checked whether a polygon is searchable, without any interpretation and explanation regarding the reason behind the searchability. For representative work along this line, see [55, 67, 68, 99, 101]. Kameda *et al.* [59] recently proposed a generalized form of the VOD and SVOD of a polygon to deal with the on-line polygon search problem by a boundary 1-searcher[1].

In our approach, we make use of the advantages of both lines of approach. We rely on the generalized form of the VOD and SVOD of a polygon (See Chapter 2 for the VOD and SVOD of a polygon and see [59] for their generalized form.).

Furthermore, we relate searchable polygons and LR-visible polygons. We also discuss the relationship between a searchable polygon and a searchable room.

4.2 Visibility diagram of a polygon

We call the above-diagonal part of the VOD T and below-diagonal part B. By using T and B, we construct a diagram that can be used to facilitate our analysis of the 1-searchability problem of a polygon, as shown in the following.

[1]Under review by *IEEE Transactions on Robotics.*

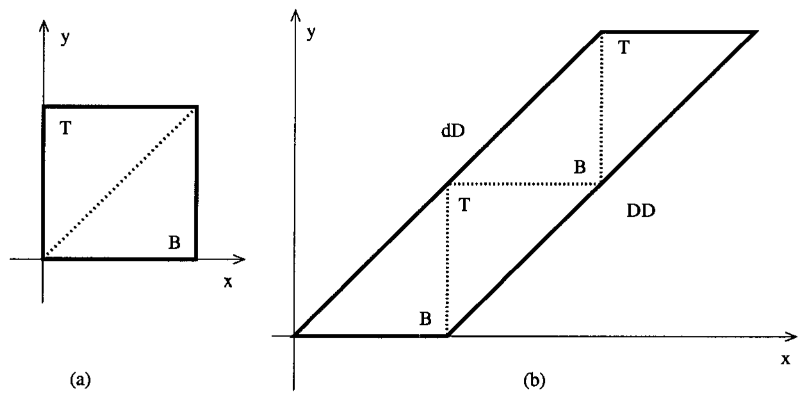


Figure 4.1: The original VOD and the construction of a new diagram.

We move T to the right of B and generate another copy of T right below B. Then we copy B to the left of the second copy of T. Thus the new diagram consists of two copies of T and B. We show the construction in Figure 4.1. In the figure, we also show the standard coordinate system. We use the *visibility space* and *visibility diagram* introduced by Kameda *et al.* [59].

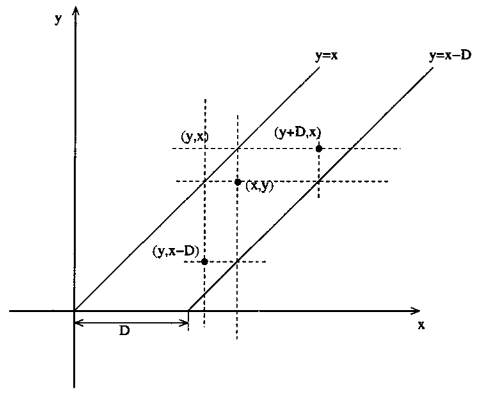


Figure 4.2: The visibility space, where $D = |\partial P|$.

We use Figure 4.2 as our reference. We call the infinite area between and including the lines $y = x$ (denoted by *dD*) and $y = x - |\partial P|$ (denoted by *DD*) the *visibility space*. Thus, point (x, y) is in the visibility space if and only if $x - |\partial P| \leq y \leq x$.

Consider a real number p. We use p to represent the point on ∂P which is at distance $p - k|\partial P|$ from the starting point p_0, where k is an integer such that $0 \leq p - k|\partial P| < |\partial P|$. This is due to the circularity of ∂P. Note that there is an infinite number of p's that can represent a single point on ∂P.

For a polygon P, we create the *visibility diagram* (VD) of P as follows. A point (x, y) in the visibility diagram is colored gray if points x and y on ∂P are not mutually visible, i.e., $\langle x, y \rangle$ is an invisible configuration.

Let a point $n_1 = (x, y)$ be in the visibility space. Consider the points $n_2 = (y, x - |\partial P|)$ and $n_3 = (y + |\partial P|, x)$. According to the above discussions, x and $x - |\partial P|$ refer to the same point on ∂P and so do y and $y + |\partial P|$. Thus, point n_1 is gray if and only if points n_2 and n_3 are gray. Their relationship is shown in Figure 4.2 [59].

Compare Figure 4.1 (b) and Figure 4.2. The points in the barriers in the former have the same relationships among the points in the latter. It is easy to see that the new diagram we constructed in Figure 4.1 (b) is a subpart of the VD of P, which extends infinitely to the lower-left and the upper-right.

For the southeast barrier due to a reflex vertex r, represented by a real number p (the distance from the origin of the polygon), the same barrier appears at regular intervals, touching dD at $(p + k|\partial P|, p + k|\partial P|)$, where k is an integer. Similarly, the northwest barrier due to r appears at regular intervals, touching DD at $(p + (k+1)|\partial P|, p + k|\partial P|)$.

A path from dD to DD in the VD of a polygon is said to be a *legal path* if it goes through the white area, except for zero or more horizontal crossings over some barriers from right to left. The following theorem directly follows from Proposition 2.3 in [68]. Also see [59].

Theorem 4.2.1 *A polygon is 1-searchable if and only if there exists a legal path in its VD.*

We then consider the skeleton representation called *skeletal visibility diagram* (SVD) of the VD. SVD is to VD as what SVOD is to VOD. We focus our analysis in the SVD.

We first introduce some notation for the SVD for our later discussions. Note that we can identify the position of a searcher on the y axis while the beam head of her flashlight on ∂P is projected on the x axis.

The reflex vertices give rise to the bones in the SVD, which are either horizontal or vertical. From dD, the bones due to a reflex vertex r extend downward and rightward and are denoted as $dDBone(r)$ and $dRBone(r)$, respectively. These bones correspond to the southeast barrier from r. From DD, the bones due to r extend upward and leftward and are denoted as $DUBone(r)$ and $DLBone(r)$, respectively. These bones correspond to the northwest barrier from r.

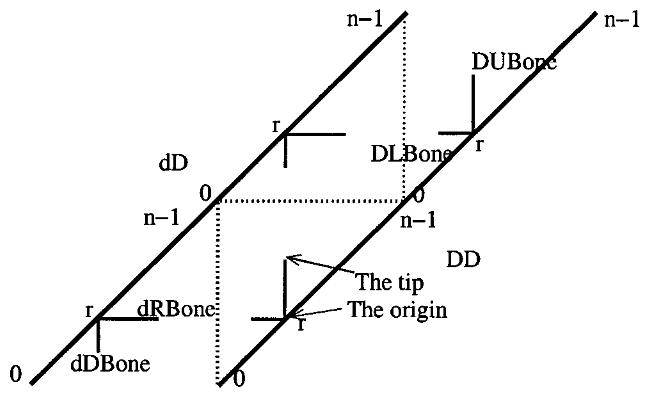


Figure 4.3: The illustration of the notation.

For each bone, we call one of its two ends that is not attached to *dD* or *DD* its *tip* while we call the other end that is attached to *dD* or *DD* its *origin.*

We call a path from *dD* to *DD* that does not cross any horizontal bones and any vertical bones from left to right the *legal path* (by abusing the notation a little) in the SVD.

See the discussions in Sections 2.3 and 2.4. Also see [67, 68] for a proof for the following theorem.

Theorem 4.2.2 *A polygon is 1-searchable if and only if there exists a legal path in its SVD.*

4.3 The searchability of a polygon

We consider legal paths in the SVD of a polygon. Throughout the search, we maintain the condition that the polygonal portion, which is on the left side of the flashlight beam in the direction from the position of the searcher to the beam head of her flashlight, is always clear. This is called the *left invariance* [67, 68]. The general idea of searching a polygon by a 1-searcher is to gradually increase the cleared portion of the polygon (though the cleared portion might temporarily decrease due to recontaminations and the backtracks of the searcher or the beam head of her flashlight. See below.) until the contaminated portion disappears. The searcher's flashlight beam divides the cleared and contaminated portions in the polygon, maintaining the left invariance.

The move of the searcher must be continuous since she always stays on the boundary. On the other hand, the move of the beam head of her flashlight can be piecewise continuous, as long as the left invariance is not violated. The only possible situation that may cause discontinuity of the beam head is where the flashlight beam hits a reflex vertex due to the rotation of the flashlight or the movement of the searcher. This situation is shown in Figure 2.10, where the beam head jumps from point 3 to point 2. When this is interpreted in the VD (marked by the dashed line segment a in the VOD), it is a crossing of a barrier from right to left. (It is the crossing of a vertical bone from right to left in the corresponding SVD.) However, since this jump maintains the left invariance, it is still allowed. On the other hand, the jump from point 2 to point 3 is not allowed since the left invariance is violated. Also note that the beam head might jump as well when the searcher moves.

When we interpret the valid moves of the searcher and the beam head of her flashlight in the SVD, it is easy to see that a path cannot cross a horizontal bone since this means the move of the searcher is not continuous. Also a path cannot cross a vertical bone from left to right. The only valid crossing happens when the path crosses a vertical bone from right to left.

4.3.1 Necessary conditions for searchability

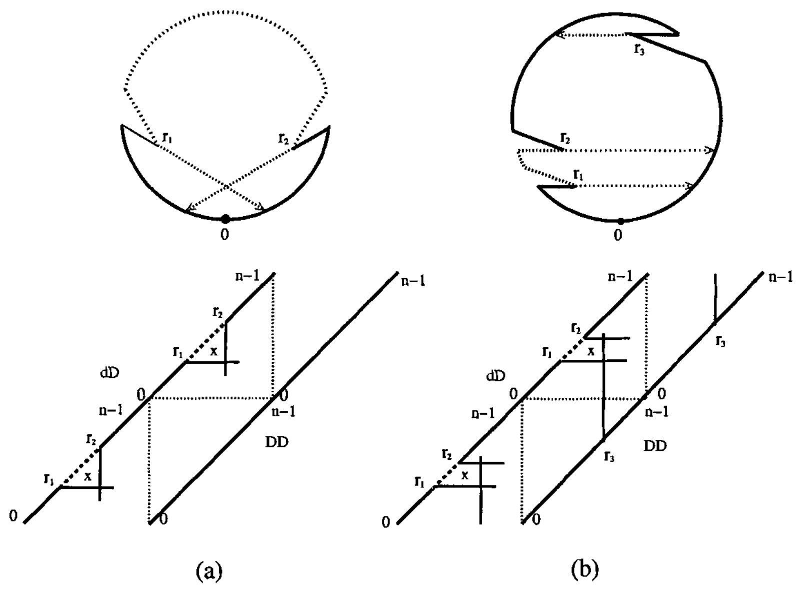


Figure 4.4: Patterns that generate trap regions bordered by dD.

We are interested in the patterns formed by different bones in the SVD. We list in Figure 4.4 the possible patterns that bound some regions bordered by dD. The pattern in Figure 4.4 (a) (The rays need not intersect.) bounds a section on $\partial P_{cw}(r_1, r_2)$ on dD (shown as the dashed line segments). The pattern also has effects on *DD*, which will be discussed shortly. It is also easy to see that the pattern in Figure 4.4 (b) (The rays from r_1 and r_2 can intersect.) bounds a section on $\partial P_{cw}(r_1, r_2)$ on *dD* (shown as the dashed line segments). The bounded regions are marked by x in the figure. We call the regions marked by x the *trap regions* on *dD*.

If a point s on *dD* in the SVD is inside a trap region, then any path starting from s cannot be extended beyond the region without violating the left invariance requirement. We immediately know that the searcher should not start a search from such a point. If this situation happens, we say that s is *trapped* on *dD*. If every point on *dD* is trapped, we say that *dD* is *trapped*. It is easy to see that if *dD* is trapped, the searcher has nowhere to start a search, indicating that the polygon is not searchable. We have the following proposition.

Proposition 4.3.1 *If dD of the SVD of a polygon P is trapped, then P is not 1-searchable.*

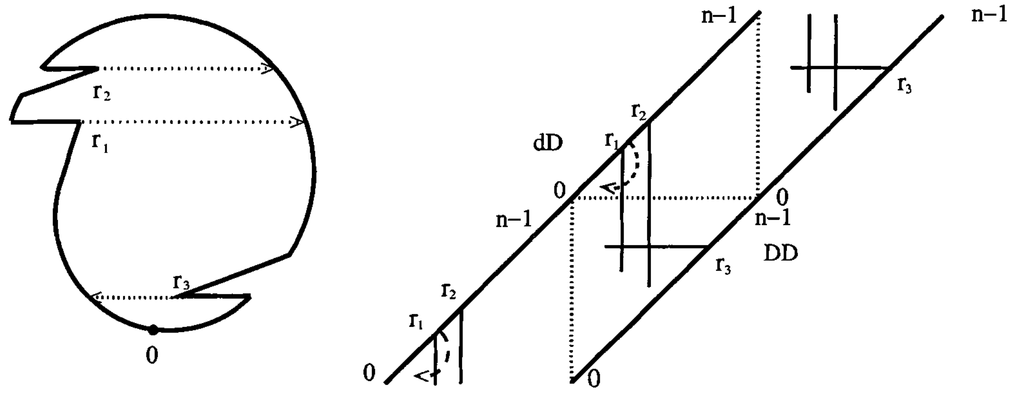


Figure 4.5: A pattern that does not make any point trapped on *dD*.

It should be noted the symmetric pattern of Figure 4.4 (b) shown in Figure 4.5 does not make any point trapped on *dD*, since a path can go across the vertical bone due to r_1 from right to left, as shown by the dashed curves in the figure.

Similarly, we list all the possible patterns in Figure 4.6 that make some regions bordering with *DD* inaccessible. The bounded regions are marked by x in the figure. The two rays in Figure 4.6 (a) need not intersect while the rays from r_2 and r_3 in Figure 4.6 (b) can intersect. (Note that the pattern in Figure 4.6 (b) is topologically identical to the one in Figure 4.5.) We call such regions *unreachable regions*. If a point t on *DD* is inside an unreachable region,

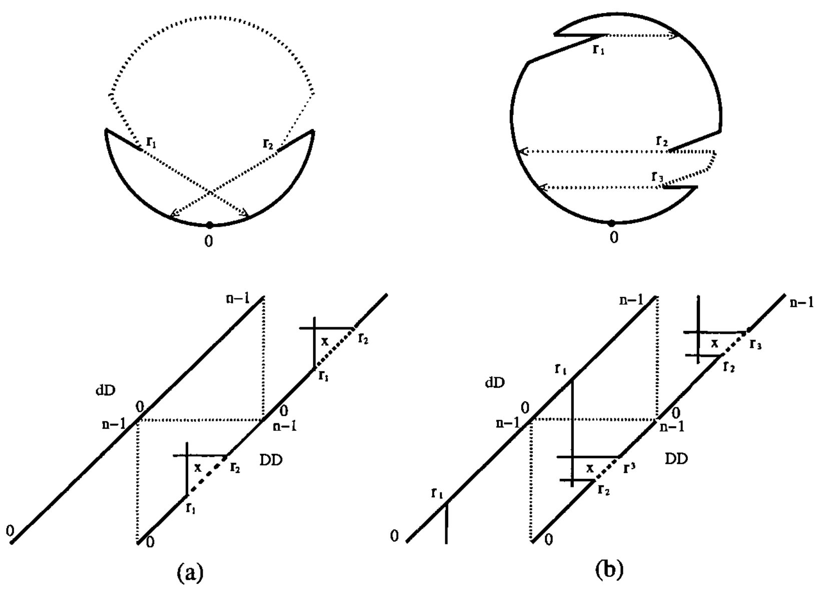


Figure 4.6: Patterns that generate unreachable regions bordered by *DD*.

then a path cannot reach *t* without violating the left invariance requirement. In such a case, we say that *t* is *unreachable* on *DD*.

If every point on *DD* is unreachable, then *DD* is said to be *unreachable*. If a point on *DD* is not unreachable, we say that it is *reachable*. We immediately have the following.

Proposition 4.3.2 *If DD of the SVD of a polygon P is unreachable, then P is not 1-searchable.*

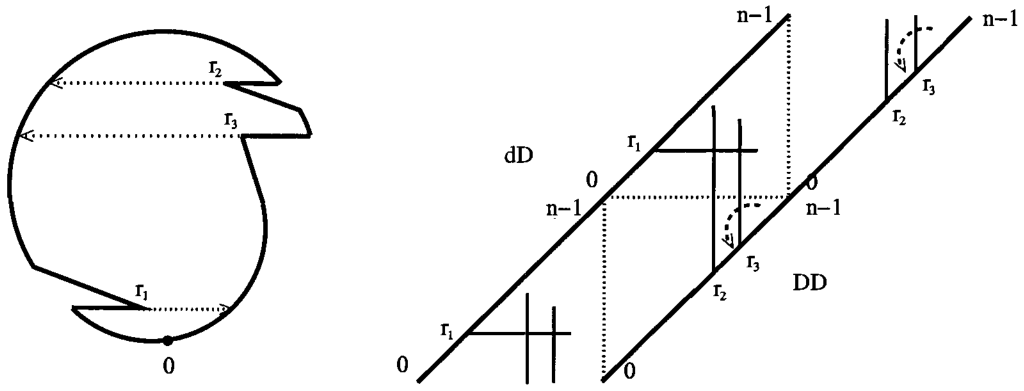


Figure 4.7: A pattern that does not make any points on *DD* unreachable.

Note that the pattern shown in Figure 4.7, which is topologically identical to the one in Figure 4.4 (b), does not make any points on DD unreachable.

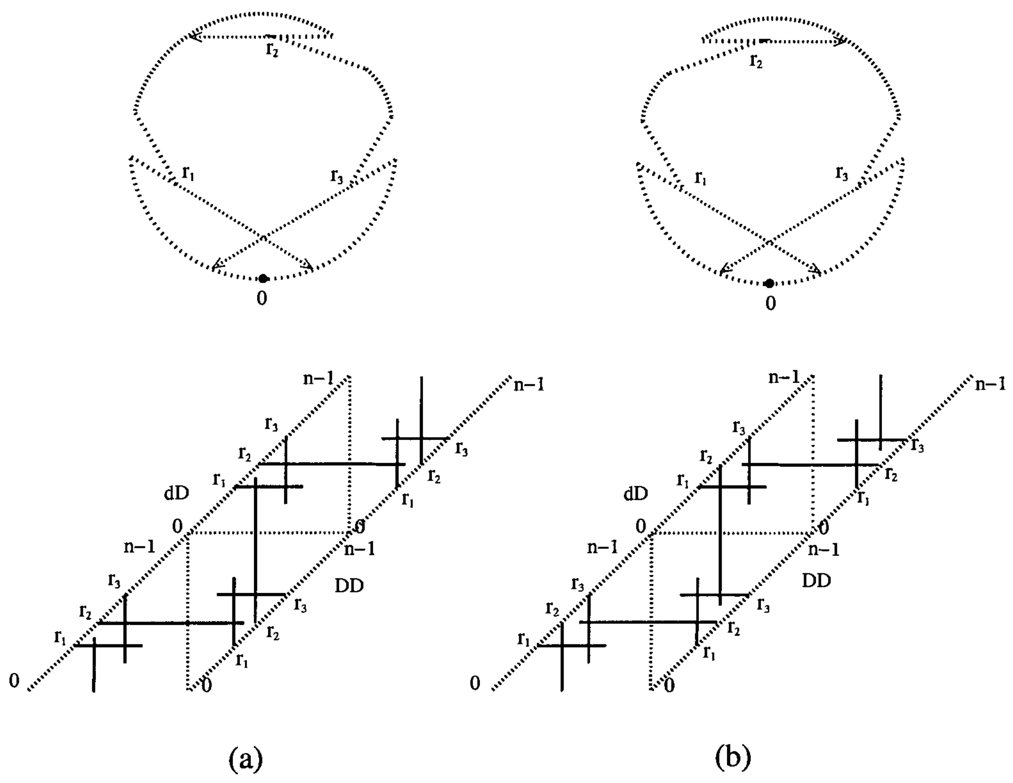


Figure 4.8: Two special patterns that make a polygon non 1-searchable.

In addition, we have two special patterns [91, 92, 106, 109] that are not included either in Figure 4.4 or in Figure 4.6. They are shown in Figure 4.8. Note that the rays from r_1 and r_3 need not intersect. It is obvious from the VDs shown in Figure 4.8 that, if the pattern shown in Figure 4.8 (a) or (b) is present in a polygon P, then P is not 1-searchable due to Theorem 4.2.2.

Lemma 4.3.3 *In the SVD of a polygon P, if every point on dD is between two reflex vertices such as r_1 and r_2 in the pattern shown in Figure 4.5, P is not 1-searchable.*

Proof The counterpart of the pattern shown in Figure 4.5 on DD is the pattern shown in Figure 4.6 (b). Thus it is easy to see that if every point on dD is between two reflex vertices such as r_1 and r_2 in the pattern shown in Figure 4.5, then every point on DD is unreachable. The lemma then follows. ▮

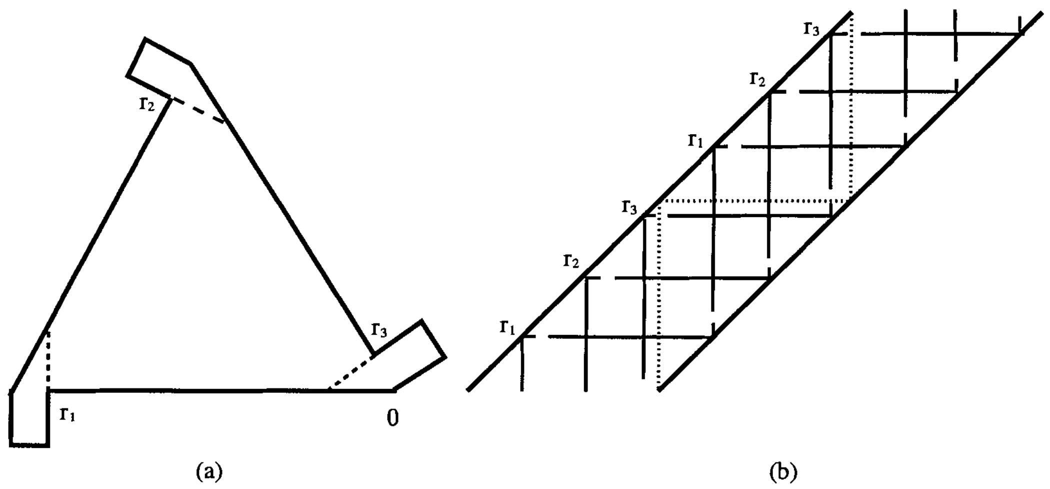


Figure 4.9: A non-searchable polygon (due to Lemma 4.3.3) and its corresponding SVD.

For an example of the situation described in Lemma 4.3.3, see the polygon and its corresponding SVD in Figure 4.9.

On the other hand, if in the SVD of a polygon P every point on *DD* is between two reflex vertices such as r_1 and r_2 in the pattern shown in Figure 4.7, P is not 1-searchable either. This can be explained as follows. The counterpart of the pattern shown in Figure 4.7 on *dD* is the pattern shown in Figure 4.4 (b). Thus it is easy to see that if every point on *DD* is between two reflex vertices r_1 and r_2 as shown in Figure 4.7, then every point on *dD* is trapped. As an example, the polygon shown in Figure 4.10 is not 1-searchable due to this reason.

4.3.2 Sufficient conditions for searchability

In order to test the 1-searchability of a polygon P, we first introduce some definitions related to the SVD of P. In the SVD, we call a maximal connected white area a *cell.* Due to the circularity of the boundary of a polygon, a cell may have an infinite area.

We show in Figure 4.11 a polygon (adopted from [59]) and its corresponding SVD with several cells indicated (surrounded by dashed line segments).

A cell *touches dD* (*DD*, resp.) if it is bordered by *dD* (*DD*, resp.). For instance, cell A in Figure 4.11 touches *DD* while cell C touches *dD.*

If two bones intersect as shown in Figure 4.12, we say that the two bones form a *bridge*, in the sense that they connect *dD* and *DD.* It is easy to see that if there is no bridge in

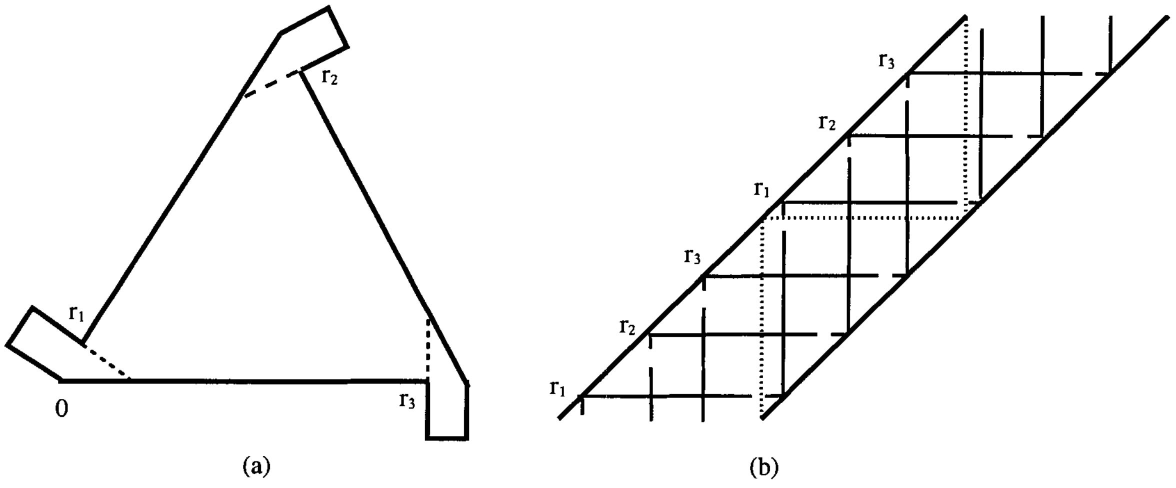


Figure 4.10: Another special non-searchable polygon and its corresponding SVD.

the SVD of polygon, then there is an infinite cell in between *dD* and *DD*. For instance, the bones from vertices 5 and 12 in Figure 4.11 form a bridge.

Figure 4.13 was obtained from Figure 4.11 by changing the position of vertex 6 slightly to remove the bridges.

Lemma 4.3.4 *If neither dD is trapped nor DD is unreachable and there is no bridge in the SVD of a polygon P, then P is 1-searchable.*

Proof It is easy to see that if there is no bridge in the SVD, the patterns shown in Figure 4.8 cannot be present in the polygon. Also the condition in Lemma 4.3.3 cannot hold.

We select a point s on *dD* that is not trapped and a point t on *DD* that is reachable. We construct a legal path connecting them.

The construction consists of two stages. In the first stage, we keep extending the path horizontally from s. In this process, the path might encounter some downward bone from *dD*. If this is the case, the path goes along the bone downward until it reaches the tip of the downward bone. This should be possible since any rightward bone from *dD* preventing the path from doing so would generate a trap region on *dD* that s is in. Note that we need not consider leftward bones from *DD*, since such a bone would imply a bridge. When the path is at the tip of the downward bone, we keep extending it horizontally, until it reaches the next downward bone from *dD*, if any, where we continue the above process, or it reaches an upward bone from *DD*, where we switch to the next stage, or it reaches a point on *DD*, in

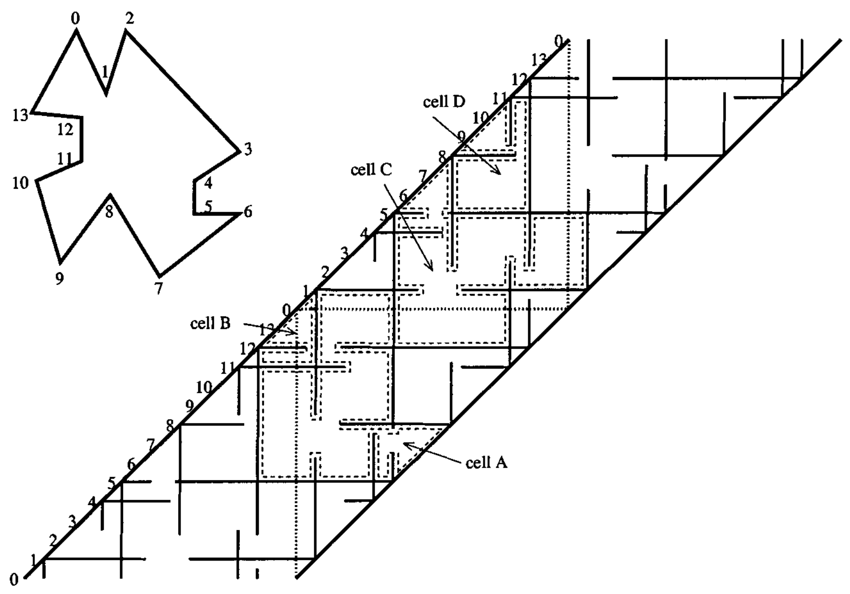


Figure 4.11: A polygon and its corresponding SVD with some cells indicated.

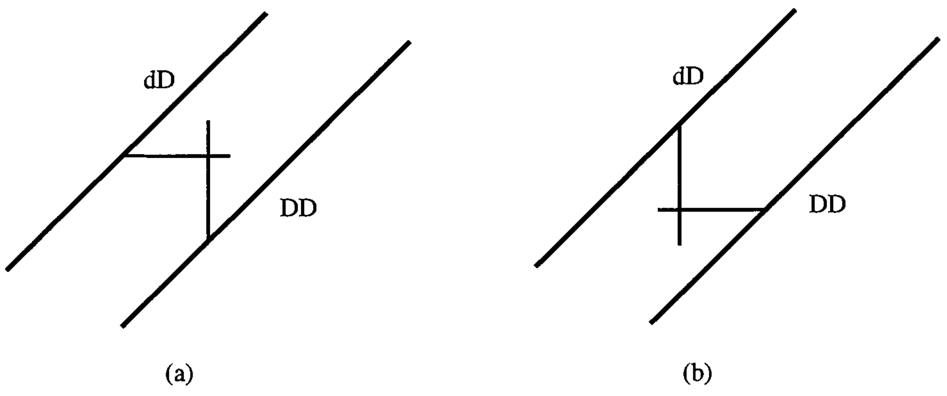


Figure 4.12: Bridges connecting *dD* and *DD*.

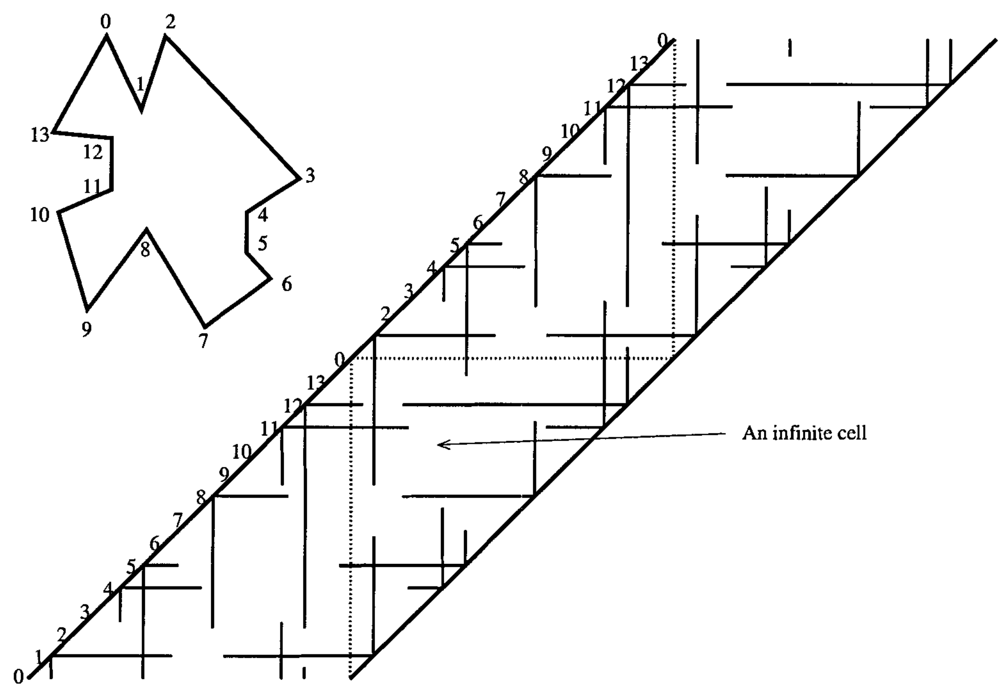


Figure 4.13: A polygon and its corresponding SVD with all the bridges removed.

which case the proof is finished by setting t to this point.

In the second stage, the path is initially close to an upward bone from DD. Then we extend the path along the bones from DD. Since there is no bridge, this extension is always possible. Eventually, the path can reach t. The lemma thus follows. ▮

Figure 4.13 shows an example in which there is no bridge in the SVD of a polygon. We show in Figure 4.14 a legal path as the dashed curve.

Before proving the following lemma, we identify three types of cells. The first type of cell has either a non-trapped point on dD or a reachable point on DD on its boundary. We call this type of cell *live cell.* The second type of cell touches either dD or DD but has neither a non-trapped point on dD nor a reachable point on DD on its boundary. We call them *dead cells.* The third type of cell touches neither dD nor DD. We call them *neutral cells.* Figure 4.15 shows neutral cells that a path from dD to DD might go through. Note that there are other neutral cells, but a path never goes through them. One such neutral cell is shown in Figure 4.16.

Note that the boundary of a cell is composed of the bones, dD and DD in the SVD.

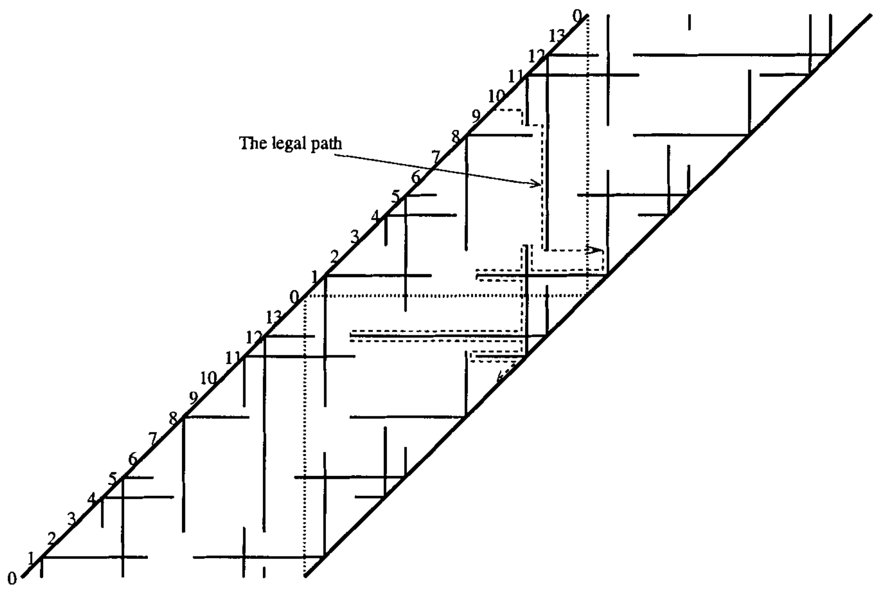


Figure 4.14: A legal path in the SVD in Figure 4.13.

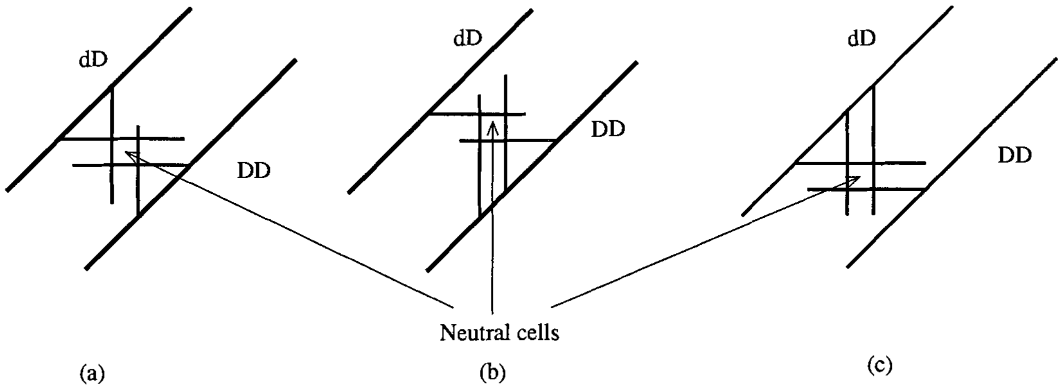


Figure 4.15: The neutral cells that a path from *dD* to *DD* might go through.

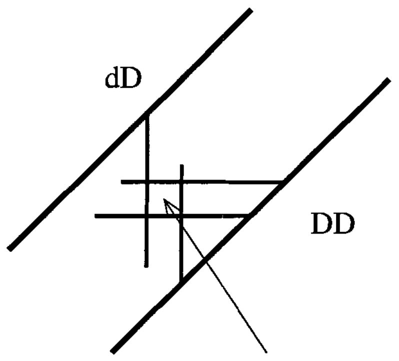


Figure 4.16: A neutral cell that a path from *dD* to *DD* never goes through.

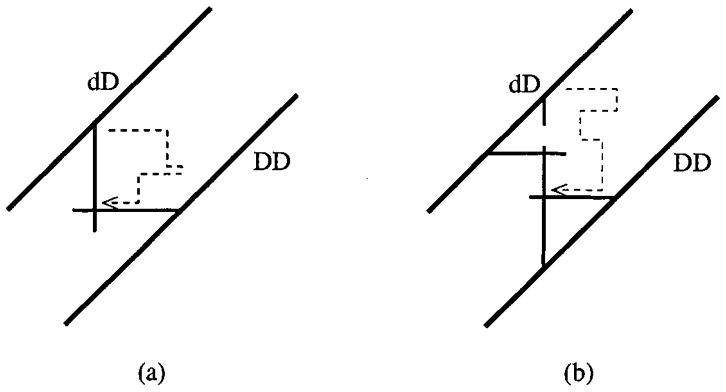


Figure 4.17: The situations where the path hits a vertical bone.

Lemma 4.3.5 *If neither dD is trapped nor DD is unreachable and the patterns in Figure 4.8 are not present in the SVD of a polygon P, P is 1-searchable.*

Proof If there is a cell touching both *dD* and *DD* and a non-trapped point *s* on *dD* and a reachable point *t* on *DD* are both on the cell's boundary, we can obviously construct a path from *s* to *t* and the lemma follows.

In order to construct a path under the conditions specified in the lemma, we select a point *s* on *dD* that is on the boundary of a live cell. We extend the path inside the cell by following the cell boundary in the clockwise direction (the cell is always on the right side of the path), until it reaches a place where the intersecting bones (part of the cell's boundary) form a bridge, as shown in Figure 4.17. The path can go across the vertical bone from right to left, since the crossing is valid.

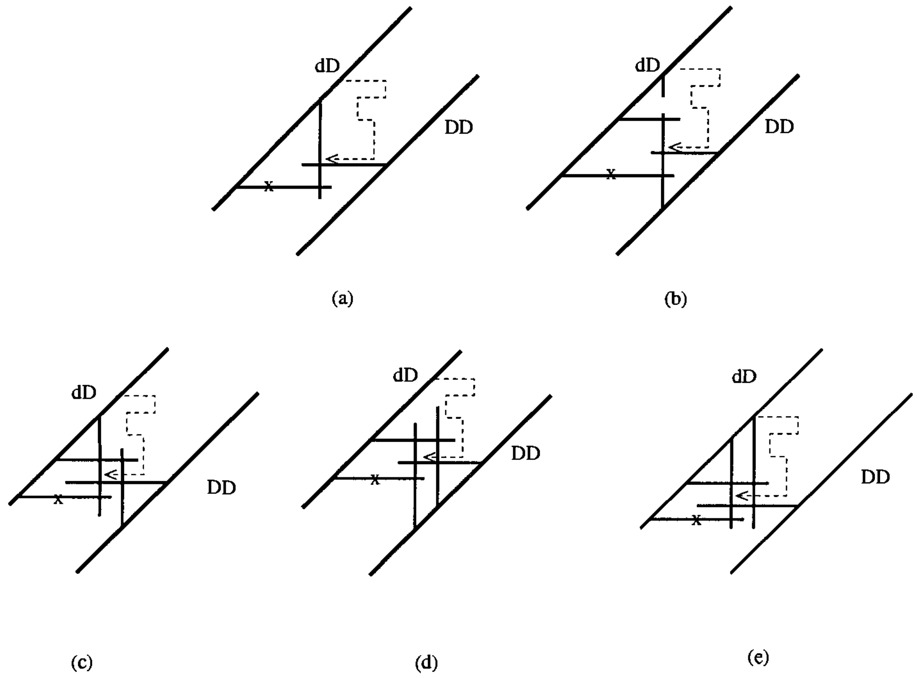


Figure 4.18: The situations where the path goes into a dead cell.

After this, the path can go through some live or neutral cells. However, after going through those "good cells", the path cannot be in a dead cell.

We list all the possible situations in Figure 4.18 where the last cell the path went through is a live one (Figures 4.18 (a) and (b)) or a neutral one (in Figure 4.18 (c), (d) and (e)). Note that the SVOD might contain other bones. But we are more interested in the bones shown in the figure.

We first consider the situations in Figures 4.18 (a), (c) and (e). If we have a dead cell ahead of the path (due to the presence of the bone marked by x), we have the special pattern shown in Figure 4.8 (b). As for the situations in Figures 4.18 (b) and (d), if we have a dead cell (due to the presence of the bone marked by x) that the path hits, we have the special pattern shown in Figure 4.8 (a).

In all the situations, if the path goes into a dead cell, we have a contradiction with the conditions specified in the lemma. Thus, when the path goes across a vertical bone from right to left, the path is always in a live cell or a neutral cell.

In this process, if the path goes into a live cell touching DD, the lemma then follows. Otherwise, we keep extending the path inside the new live cell by following its boundary in

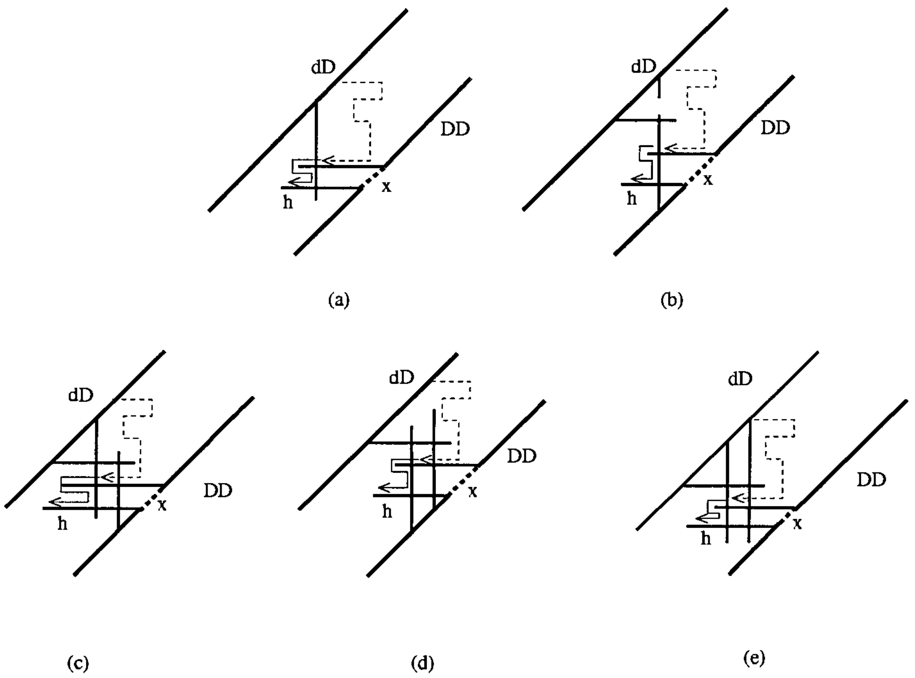


Figure 4.19: The situations where the path hits a leftward bone from *DD*.

the clockwise direction.

Now if the path is in a live cell touching *dD*, we may have one of the situations in Figure 4.19. The path continues its extension but hits a leftward bone (part of the cell boundary) (marked by *h*) from *DD*. It then has to follow the bone leftward. However, in each situation, this bone and other bones together generate an unreachable region bordered by *DD*, as marked by x (the dashed line segments) in the figure. Note that there might be other unreachable regions bordered by *DD*. But we only show those related to the leftward bone. Also note that there might be other bones in the SVOD, but we are more concerned about whether the SVOD contains bones causing one of the situations in Figure 4.19.

The path further extends leftward until it hits a bridge, in which case the path is again in one of the situations shown in Figure 4.18. We know that the path should be still alive and can be further extended. If no matter how the path is extended, it is always in a live cell touching *dD* (after going through zero or more neutral cells), then according to the above discussions, the access to *DD* is always hindered by an unreachable region bordered by *DD* in this process. Assume that the path keeps extending to the lower left indefinitely. Imagine that the sections of the vertical bones that the path goes through were absent.

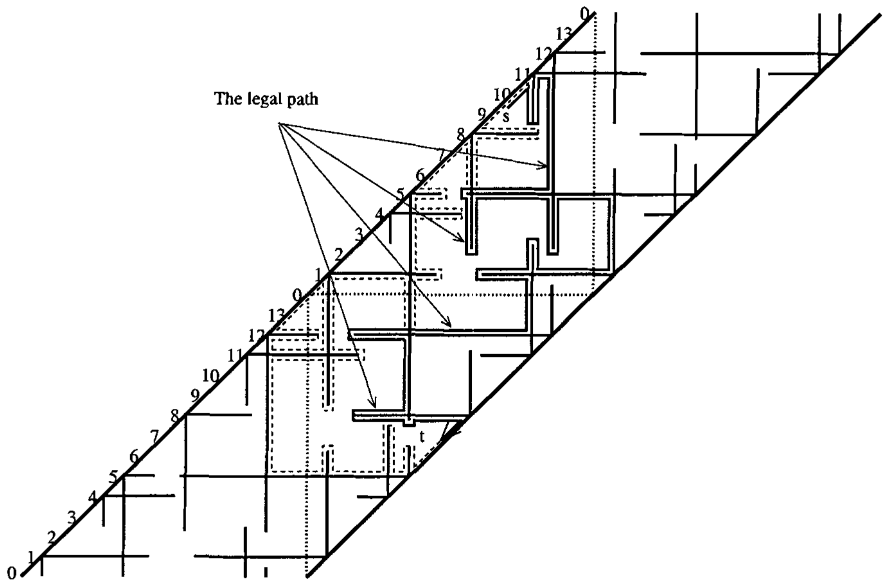


Figure 4.20: A legal path in the SVD in Figure 4.11.

Then it is as if the path were going through an infinite cell, but can never reach *DD*. This could only mean that there are unreachable regions spanning the entire *DD*, which leads to a contradiction. For instance, the situation described by Lemma 4.3.3 shows such a case.

We conclude that the path should be able to reach a live cell touching *DD*. The lemma thus follows. ▮

Figure 4.11 shows an example in which there are bridges in the SVD of a polygon. We show in Figure 4.20 the legal path constructed. It is indicated by the solid, arrowed curve directed from s to t.

Theorem 4.3.6 *A polygon P is 1-searchable if and only if in the SVD of P neither dD is trapped nor DD is unreachable and the patterns in Figure 4.8 are not present.*

4.4 Searchability and LR-visibility

In Chapter 3, we proved that a searchable room must be LR-visible. In this section, we prove that a searchable polygon is LR-visible as well.

We first give the following lemma [36].

Lemma 4.4.1 *If a polygon contains three disjoint components, then it is not LR-visible.*

See Section 1.2 for the definition of a component and a non-redundant component in a polygon. The following lemma was first proved in [106].

Lemma 4.4.2 *If a polygon contains three disjoint components, then it is not 1-searchable.*

Proof The lemma follows due to either Lemmas 4.3.3 or that we have one of those special patterns shown in Figure 4.8 in the polygon. ▮

By Lemma 4.4.2, there are only two cases of our interest in a searchable polygon; either there are no disjoint components or there is just a pair of disjoint components.

We first consider the situation where all the components in a searchable polygon have a common intersection.

Lemma 4.4.3 *If the components in a searchable polygon have a common intersection, the polygon is LR-visible.*

Proof We can select two points $s \prec_{cw} t$ on the common boundary of all the components. We immediately know that the polygon is LR-visible with respect to s and t since every component contains both of them [36]. ▮

We now only need to consider a searchable polygon that contains two disjoint components, as shown in the following series of lemmas.

Lemma 4.4.4 *If a 1-searchable polygon P contains only a pair of disjoint components, one of which is clockwise while the other is counter-clockwise, P is LR-visible.*

Proof we try to locate two points s and t on ∂P such that P is LR-visible with respect to s and t.

Assume that the clockwise component is due to a reflex vertex a and the counter-clockwise component is due to a reflex vertex b, as shown in Figure 4.21.

We first consider the components that only intersect with the component due to a. It is easy to see that the intersection of these components is a continuous boundary inside the component. We call it T.

There can be only three situations here, which are shown in Figure 4.22. In the figure, we only show the two components that generate the starting point and the ending point of

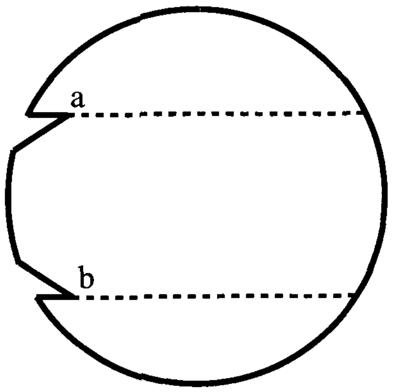


Figure 4.21: Two disjoint components. (The clockwise component is due to vertex a while the counter-clockwise component is due to vertex b.)

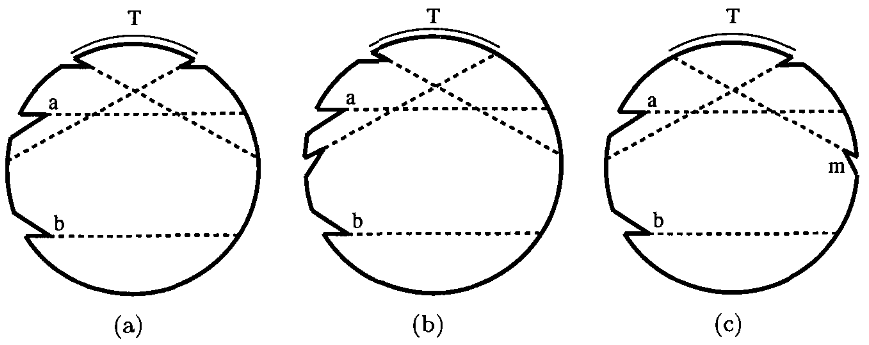


Figure 4.22: Definition of T for for the components in Figure 4.21.

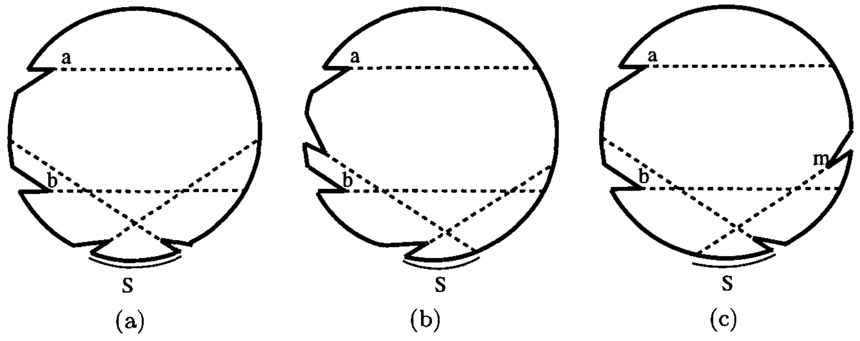


Figure 4.23: Definition of S for the components in Figure 4.21.

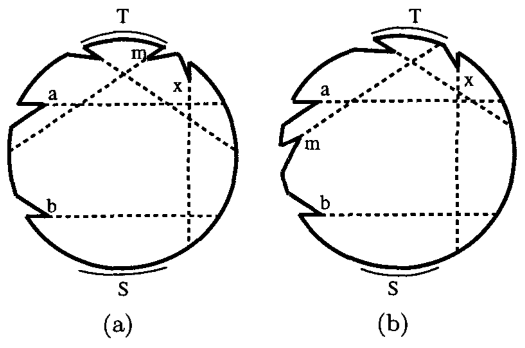


Figure 4.24: The situation where the clockwise component due to x does not intersect T or S, and is between T and S in the clockwise direction.

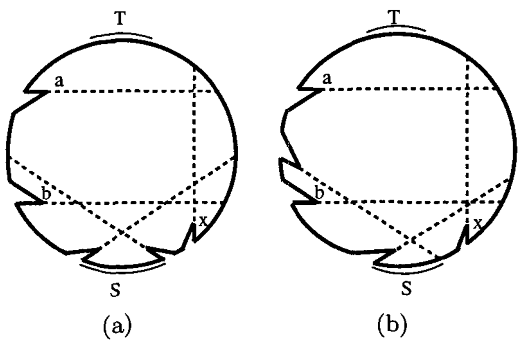


Figure 4.25: The situation where the counter-clockwise component due to x does not intersect T or S, and is between T and S in the clockwise direction.

T. However, the situation shown in Figure 4.22 (c) is the special pattern shown in Figure 4.8 (a), with $m = r_1$, $b = r_2$, and $a = r_3$. We thus drop it from later discussions.

Similarly, we consider the components that only intersect with the component due to b. We also have an intersection of those components, which is within the component and is denoted as S. Figure 4.23 shows three situations. Again, the situation in Figure 4.23 (c) is the special pattern shown in Figure 4.8, with $b = r_1$, $a = r_2$, and $m = r_3$. We thus drop it from later discussions as well.

The only components left are those that intersect with both the component due to a and the component due to b. However, we claim that such a component must intersect (contain) S, T or both, as shown in the following.

We assume that the component under discussion is due to a reflex vertex x in the following discussions. We consider several situations.

We first consider that the component due to x is clockwise and is between T to S in the clockwise direction. Suppose that the component from x does not intersect T or S. We list all the possible situations in Figure 4.24. It is easy to see that all these situations make the polygon non-searchable. For instance, the three components due to the reflex vertices x, m and b in Figure 4.24 (a) form the special pattern shown in Figure 4.8 (a), while the three components due to the reflex vertices x, m and b in Figure 4.24 (b) form the special pattern shown in Figure 4.8 (b)

Next we assume that the component due to x is counter-clockwise and is between T and S in the clockwise direction. For the same reason as above, it is easy to see that the two possible situations, as shown in Figure 4.25, render the polygon to be non-searchable.

We then consider the situation where the component due to x is clockwise but is between S and T in the clockwise direction. The possible situations are shown in Figure 4.26.

We show in Figure 4.27 the two possible situations for Figure 4.26 (b) where we include the components that form S. It is not difficult to see, using the same argument as above, that in any of these situations, the polygon is not searchable. A similar argument holds true for the pattern in Figure 4.26 (a).

The last possibility we need to consider is that the component due to x is counter-clockwise but is between S and T in the clockwise direction. Again we list the two possible situations in Figure 4.28. For the patterns in Figure 4.28 (a) and (b), we further consider them, just as we did above. We list the possible situations in Figure 4.29 for the pattern shown in Figure 4.28 (b) when the components forming T are considered. It is easy to see

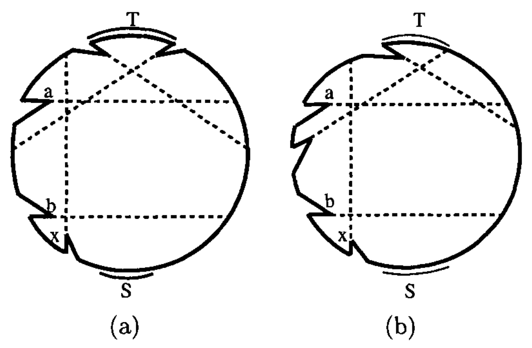


Figure 4.26: The situation where the clockwise component due to x does not intersect S or T, and is between S and T in the clockwise direction.

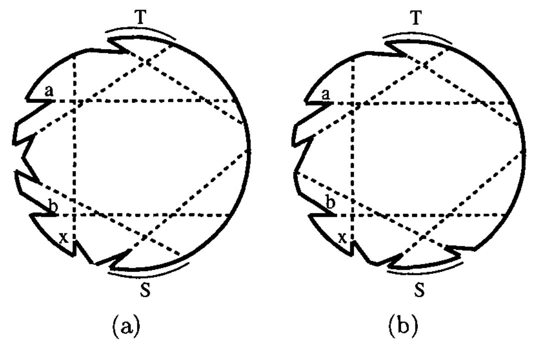


Figure 4.27: Further situations for the pattern in Figure 4.26 (b).

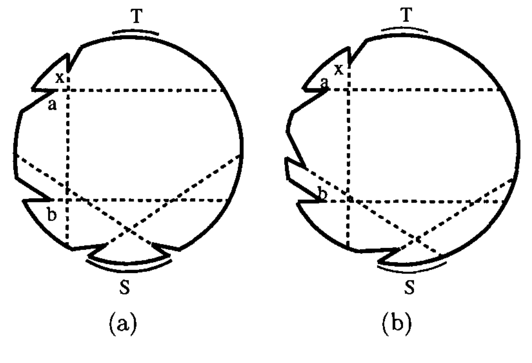


Figure 4.28: The situation where the counter-clockwise component due to x does not intersect S or T, and is between S and T in the clockwise direction.

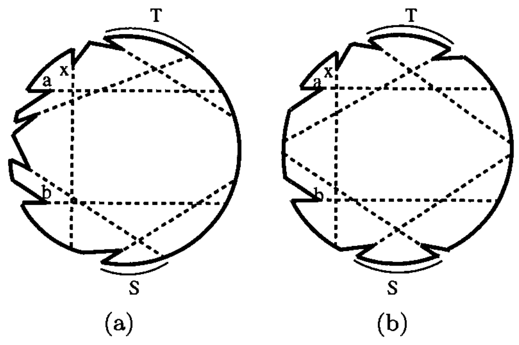


Figure 4.29: Further situations for the pattern in Figure 4.28 (b).

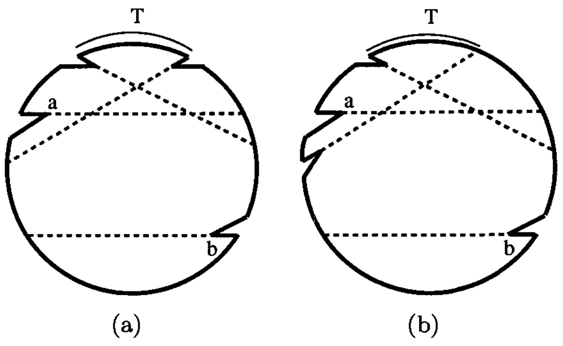


Figure 4.30: The situations for T where both the components due to a and b are clockwise.

that each of the situations makes the polygon non-searchable. The pattern in Figure 4.28 (a) is dealt with similarly.

We conclude that every component must intersect or contain S or T. Recompute the intersection for these components, and adjust S and T. After the adjustment, select a point $s \in S$ and a point $t \in T$. Every component must contain either s or t. Thus P is LR-visible with respect to s and t. The lemma follows. ▮

Lemma 4.4.5 *If a 1-searchable polygon P only contains two disjoint components that are either both clockwise or both counter-clockwise, then P is LR-visible.*

Proof We consider here the case where the two components are clockwise. The counter-clockwise case can be dealt with similarly.

We list the two situations where T is formed in Figure 4.30 when the two disjoint components due to a and b are both clockwise. The situation in Figure 4.30 (a) involves a clockwise component and a counter-clockwise component and was dealt with in Lemma 4.4.4. The only situation left is in Figure 4.30 (b).

Similarly, for S we can list the two situations in Figure 4.31 and only the situation in Figure 4.31 (b) needs to be discussed.

Thus the only combination is between the situation in Figure 4.30 (b) and the situation in Figure 4.31 (b). We consider a clockwise component between T and S in the clockwise direction, as shown in Figure 4.32. It is not searchable since every point on DD is unreachable, according to Figure 4.6 (b). Note that we need not consider any counter-clockwise component, since the situation can be dealt with by Lemma 4.4.4.

The last possibility is shown in Figure 4.33, which can also be handled similarly, since every point on DD is unreachable.

For the above two situations, also see Lemma 4.3.3 and the discussions following it.

Summarizing the above arguments, the lemma follows. ▮

Theorem 4.4.6 *If a polygon P is 1-searchable, then it is LR-visible.*

Proof It follows from Lemmas 4.4.3, 4.4.4, and 4.4.5. ▮

4.5 Checking the searchability

The main reason that we relate the searchability and LR-visibility of a polygon is that we hope to take advantage of the nice property of a LR-visible polygon, i.e., we can calculate

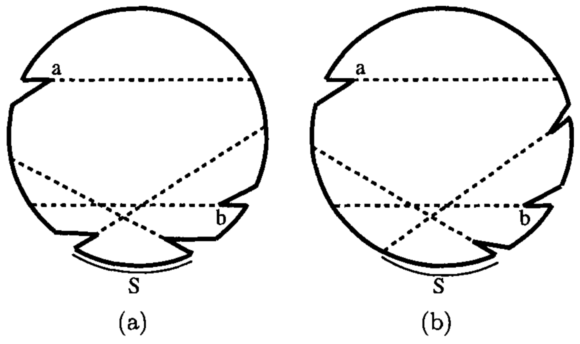


Figure 4.31: The situations for S where both the components due to a and b are clockwise.

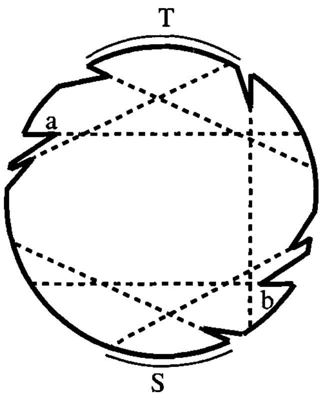


Figure 4.32: The situation where there is a clockwise component between T and S in the clockwise direction.

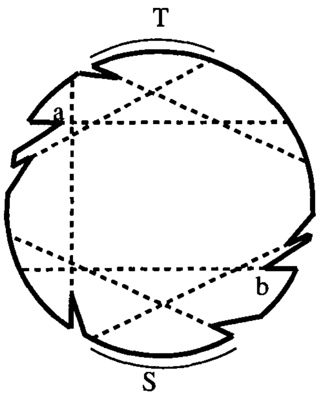


Figure 4.33: The situation where there is a clockwise component between S and T in the clockwise direction.

the non-redundant components in $O(n)$ time in such a polygon.

However, as shown in the proof of Theorem 4.3.6, we need to consider not only the non-redundant components in a polygon, but also the redundant components when checking its searchability. Even though this additional requirement might be eliminated by studying the other properties of a 1-searchable polygon, so far none of our attempts was successful.

For a polygon with n vertices, the deadlocks in a polygon can be calculated in $O(n)$ time [15] (since the polygon must be *LR*-visible), making the calculations of trapped parts on *dD* and unreachable parts *DD*, respectively, in $O(n)$ time. There can be at most $O(n)$ such parts. Also all the components (including the redundant and non-redundant ones) can be calculated in $O(nlog(n))$ time.

As for the two special patterns in Figure 4.8, we consider Figure 4.8 (a) here, since Figure 4.8 (b) can be dealt with in a similar fashion. For a deadlock (calculated as above) formed by the reflex vertices r_1 and r_3, we look for a third reflex vertex r_2, forming the pattern in Figure 4.8 (a). This can be done in $O(n^2)$ time for all the deadlocks.

As for the patterns in Figure 4.4 (b) and in Figure 4.6 (b), we discuss the situation for the pattern in Figure 4.4 (b) since the pattern in Figure 4.6 (b) can be dealt with similarly. For each reflex vertex r_1, we look for a reflex vertex r_3 whose $r_{3_{ccw}}$ is the farthest from r_1 in the clockwise direction and whose counter-clockwise component does not intersect with the one due to r_1. We have at most $O(n)$ such pairs of reflex vertices. For each pair, such as r_1 and r_3 as calculated above, going from $r_{3_{ccw}}$ in the counter-clockwise direction toward r_1, we try to find a reflex vertex r_2 which is closest to $r_{3_{ccw}}$ in the counter-clockwise direction and whose counter-clockwise component does not intersect the one due to r_3. If such a reflex vertex r_2 is found, then we have a trap region bordered by *dD*, which makes any point on $\partial P_{cw}(r_1, r_2)$ trapped on *dD*. For all the $O(n)$ pairs, clearly this can be done in $O(n^2)$ time.

Thus, in order to check the searchability using our characterization of a 1-searchable polygon, we need $O(n^2)$ time. However, we do not believe that this bound is tight. But due to the time limitation, we did not explore further on this.

Time complexity $O(nlog(n))$ for checking the searchability of a polygon was reported in [91, 92], while the work in [109] presented an algorithm for the same purpose with time complexity of $O(n^2)$.

4.6 Discussions

We have characterized the polygons that are searchable by a 1-searcher. Our work here is complementary to the one in [67, 68], where only an algorithmic graph-based solution is provided and no characterizations are discussed, but is much simpler than the ones reported in [91, 92, 109] in terms of the analysis complexity. We attribute this to the VD and SVD of a polygon.

We have noticed that the relationship between *LR*-visibility of a polygon and its 1-searchability has been previously explored in [106]. It is stated that if a simple polygon P is 1-searchable, then there exists a pair of vertices u and v of P such that P is weakly visible from the shortest path between u and v, i.e., every point on the two sides divided by u and v can be seen at least by one point on the shortest path between u and v. Then according to the discussions in [14], P is *LR*-visible with respect to u and v.

The approach employed in [106] is simpler than ours reported here. But it is harder to follow. Interested readers are referred to the original paper.

4.6.1 The complexity of testing searchability

We wanted to improve the checking time complexity of the 1-searchability of a polygon. But we were not successful. This is definitely our task in the near future.

4.6.2 The relationship between the searchability of a polygon and a room

It first appeared that a searchable polygon could contain a door such that the resultant room is searchable by two guards. But it has turned out that the conjecture is incorrect, as shown in Figure 4.34. We show the polygon and its corresponding SVD.

It is easy to see that the polygon itself is searchable by a 1-searchable since starting from a point between vertices 1 and 5 in the clockwise direction, we can construct a legal path from *dD* to *DD*, as shown in Figure 4.34 (b).

The polygon is purposely constructed in such a way that every point on *dD* is trapped except the part between vertices 1 and 5. Since the trap regions on *dD* are the same as those trap regions when we discussed the problem of finding all doors for a given polygon in Chapter 3, it follows that any door, if one exists, cannot be a point trapped on *dD*. Therefore, the only possible place for a door must be between vertices 1 and 5 in the clockwise direction.

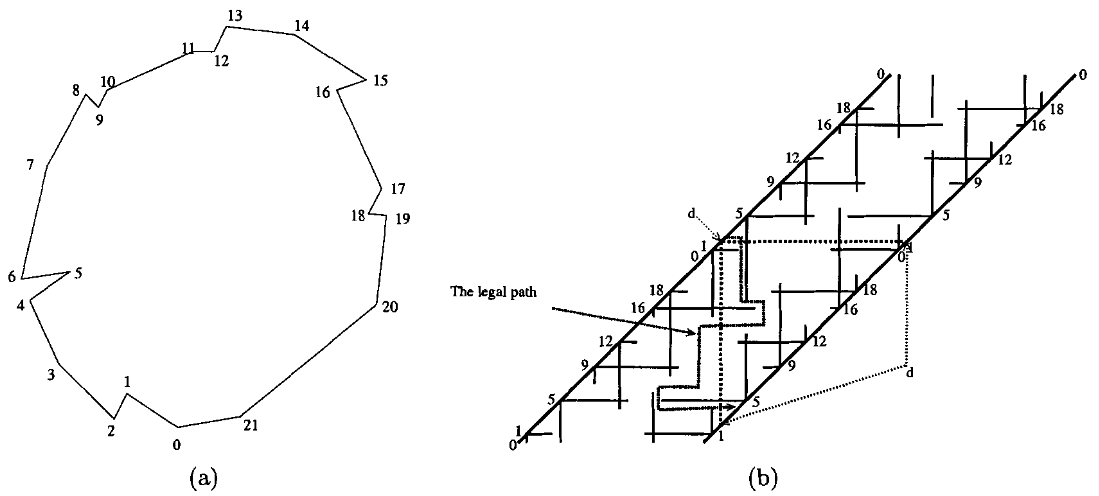


Figure 4.34: A 1-searchable polygon and its corresponding SVD.

Suppose that the door is at d, as shown in the figure. If the room $P(d)$ is searchable by two guards, it must be possible to find a path which starts at d and ends on DD within the triangle shown in the figure formed by d and its two counterparts on DD. However, we notice that every point on DD between d and d in the clockwise direction is unreachable except the points on the two parts between d and vertex 5 and between vertex 1 and d, respectively, in the clockwise direction. Even for these two parts, the points on them are unreachable on DD within the triangle because of the horizontal bone from vertex 5 and the vertical bone from vertex 1, respectively. Thus, any point between vertices 1 and 5 (including the two vertices themselves) cannot be used as a door.

It is not difficult to see that the above example can also be used to derive a similar result for the situation where the room is searched by a 1-searcher.

Chapter 5

Searching 1-Hole Polygons

The polygon search problem by two 1-searchers has been investigated by Simov *et al.* in [99], as mentioned in Chapter 1. The approach employed is algorithmic, in the sense that we can tell whether a polygon is searchable by two 1-searchers but could not interpret the reason behind this searchability since the geometric properties of the polygon are not reflected in an obvious manner in the algorithm.

In this chapter, we attempt to work on a restricted version of the polygon search problem by two boundary 1-searchers. We consider a 1-hole polygon and there are two boundary 1-searchers to search it. We will see how the extended visibility space of a 1-hole polygon provides us with hints that can be used to tackle the problem.

Without causing any ambiguity, if we say that a 1-hole polygon is searchable, it should be interpreted as that it is searchable by two boundary 1-searchers. Also a searcher should be understood as a boundary 1-searcher.

5.1 Introduction

A 1-hole polygon is a polygon with a simple polygon (the "hole") removed from its interior. The interior of the hole is considered to be exterior to the polygon itself. Therefore, there are two boundaries for such a polygon.

In order to make our discussions simple, we adopt the mutual visibility between two points used in [67, 68]. For two point $a \in \partial P$ and $b \in \partial H$, if $\overline{ab} \in P - \partial P$, then a and b are mutually visible. According to this definition, in Figure 5.1, a' and b are mutually visible

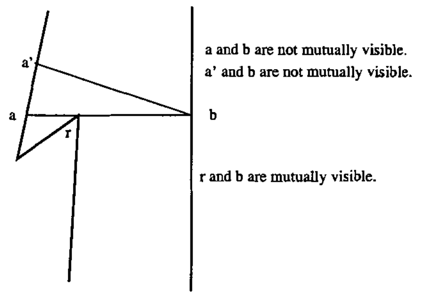

a and b are not mutually visible.
a' and b are not mutually visible.
r and b are mutually visible.
a'
a
r
b

Figure 5.1: Definition of mutual visibility [67, 68].

and so are r and b. But a and b are not mutually visible. Note a and b are mutually visible according to our previous mutual visibility definition in Chapter 1.

We require that the forward and backward ray end points from any reflex vertex be on the opposite boundary to make the problem simple. It is easy to see that under this requirement, the two boundaries are mutually weakly visible.

We also require that the two searchers stay on the boundaries. We use ∂P to refer to the outer boundary and ∂H to the inner boundary, and use a, a', a'', a_0, a_1, $\cdots$, r, r', etc. to represent vertices on ∂P, while using b, b', b'', b_0, b_1, $\cdots$, q, q', etc. to represent vertices on ∂H.

The two 1-searchers move along the boundaries to search the polygon. We consider the following three settings. (1) Both searchers move on the outer boundary; (2) Both searchers move on the inner boundary; and (3) One searcher moves on the outer boundary and the other moves on the inner boundary. In all of the three settings, it is required that the beam head of the flashlight of a searcher be always on the opposite boundary from the searcher's boundary. For simplicity, we denote by P_o, P_i, and P_b the search problems in these three settings, respectively.

5.2 The street search problem

We first consider the two-guard street search problem because it is relevant to the problem in hand. Recall that a street is defined as a polygon P with two distinguished points, s and t, called the *entrance* and *exit*, respectively, on its boundary. It is required that $\partial P_{cw}(s,t)$ and $\partial P_{ccw}(s,t)$ be mutually weakly visible. The skeletal visibility obstruction diagram (SVOD) of a street is restricted in a rectangle whose two sides represent $\partial P_{cw}(s,t)$ and $\partial P_{ccw}(s,t)$,

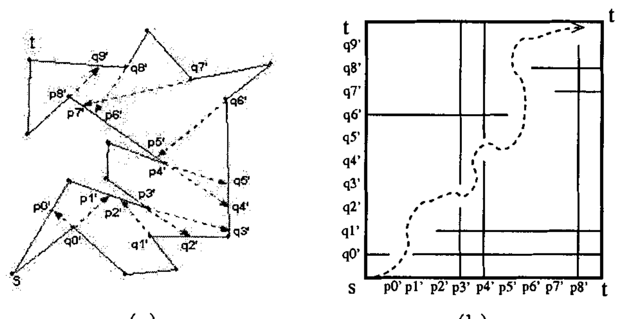


Figure 5.2: A street that is searchable by two mutually visible guards.

respectively. For example, Figure 5.2 (b) is the SVOD of the street in Figure 5.2 (a). We try to find a path through the white area which starts at s and ends at t, which means that the street is searchable by two mutually visible searchers (guards). The path in Figure 5.2 (b) is similar to the ones we introduced when we discussed the room search problem in Chapter 3 and the polygon search problem in Chapter 4.

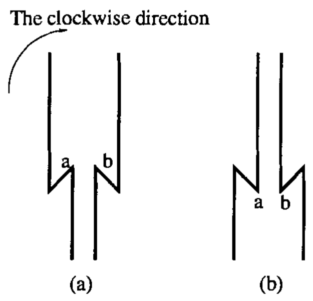


Figure 5.3: Deadlocks.

Deadlocks play an important role in determining whether a street is searchable or not. To give an illustration, we redraw the deadlocks in Figure 1.8 here in Figure 5.3. Also we draw the corresponding partial SVODs of the deadlocks in Figure 5.4. It is quite easy to see why the presence of deadlocks in a street makes it non-searchable. We still call the regions bounded by bones the *trap regions*. The path is either restricted within a trap region or cannot go into it in the SVOD.

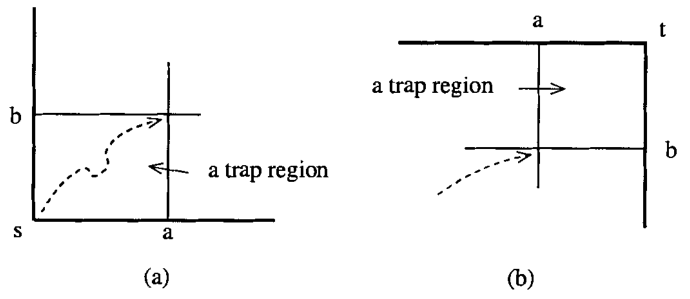


Figure 5.4: Partial SVODs corresponding to the deadlocks in Figure 5.3.

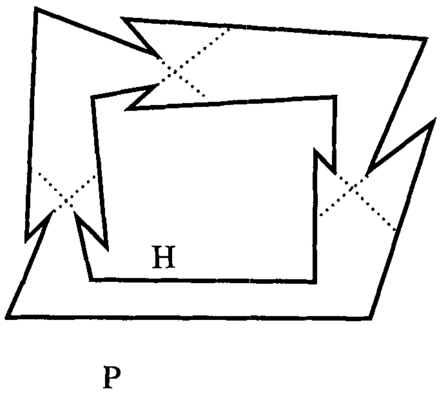


Figure 5.5: A non-searchable 1-hole polygon.

We now consider deadlocks in the 1-hole polygon search problem. As a simple example, it is easy to verify that the 1-hole polygon shown in Figure 5.5 cannot be searched by two 1-searchers, no matter where they start their search and no matter which search setting, i.e., P_i, P_o, P_b, is considered. The intuitive reason behind this is that we need the two 1-searchers together to clear the first deadlock. However, it is not possible for the two searchers to work together on the second or the third deadlock at the same time.

However, with a special arrangement of the deadlocks shown in Figure 5.6, the polygon in the figure is still searchable, even though it contains more than three deadlocks.

Since we are dealing with the search problem with two 1-searchers, intuitively we have one more freedom represented by the additional searcher to consider. We need to extend the visible space, as shown below.

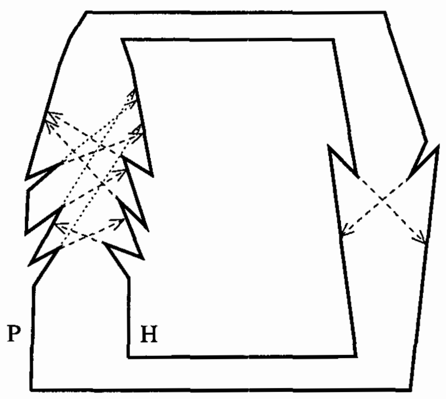


Figure 5.6: A searchable 1-hole polygon.

5.3 The visibility space of a 1-hole polygon

In order to study the 1-hole polygon search problem by two 1-searches, we employ the *visibility space* of a 1-hole polygon. It is composed of configurations $\langle r, q \rangle$, where $r \in \partial P$ and $q \in \partial H$. This is different from the visibility space we introduced in the previous chapters.

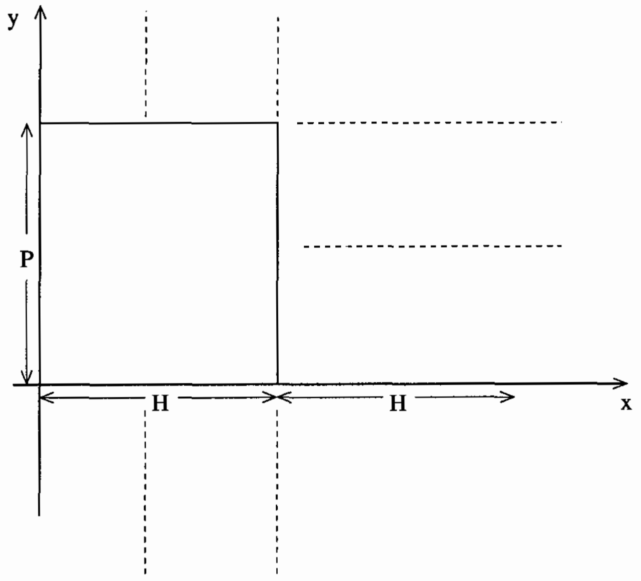


Figure 5.7: The visibility diagram of a 1-hole polygon.

In order to visualize the visibility space for a 1-hole polygon, we still use standard

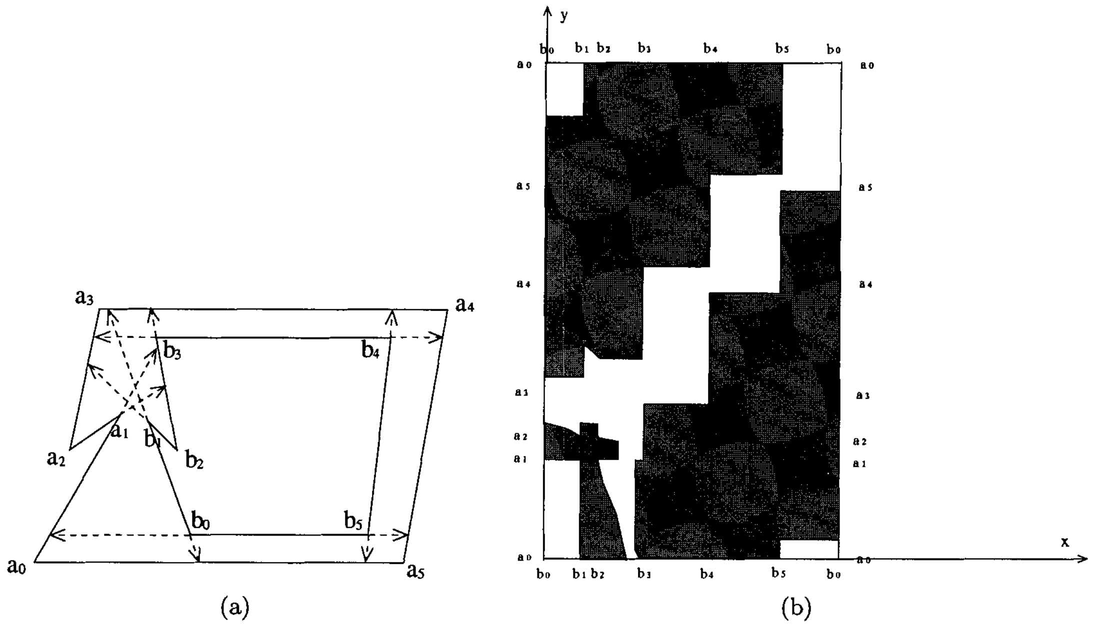


Figure 5.8: A 1-hole polygon and its VD.

coordinate system, in which the x-axis represents the clockwise traversal of ∂H while the y-axis represents the clockwise traversal of ∂P. Due to the circularity of a 1-hole polygon (like a *ring*), its visibility space extends to the whole coordinate plane. We start drawing a diagram in this plane from the point corresponding to any configuration and gray those points corresponding to the invisible configurations while leaving white those corresponding to the visible configurations. Figure 5.7 shows an example. Note that this diagram extends infinitely in all directions. We call this diagram the ***visibility diagram*** (VD) of the 1-hole polygon. We show a 1-hole polygon and its corresponding VD in Figure 5.8[1]. We use the VD to conduct our analysis.

It is easy to see that the VD of a 1-hole polygon is similar to the VOD and SVOD in Section 5.2 of a street. However, the VOD and SVOD of a street is restricted within the rectangle formed by the sides corresponding to the street's two sides, as shown in the example in Figure 5.2.

In the VD of a 1-hole polygon, a point (x, y) is white if and only if the points $(x +$

[1]Due to the time limitation, we did not create a program to draw this diagram.

$k_1|\partial H|, y + k_2|\partial P|)$ are white, where k_1 and k_2 are integers.

Consider a *visible configuration* $\langle r, q\rangle$, where $r \in \partial P$ and $q \in \partial H$. Obviously, such a visible configuration could be used to represent the standing position of a single searcher and the beam head of her flashlight. Thus, two visible configurations $\langle r_i, q_i\rangle$ $(i = 1, 2)$ together can completely determine the current states of the two searchers. In the sequel, if we say the state of a searcher, this state should be interpreted as including the standing position of the searcher and the beam head of her flashlight. Note that at this moment we do not specify rigidly which one of r_i and q_i is the standing position of a searcher and which one represents the beam head of her flashlight.

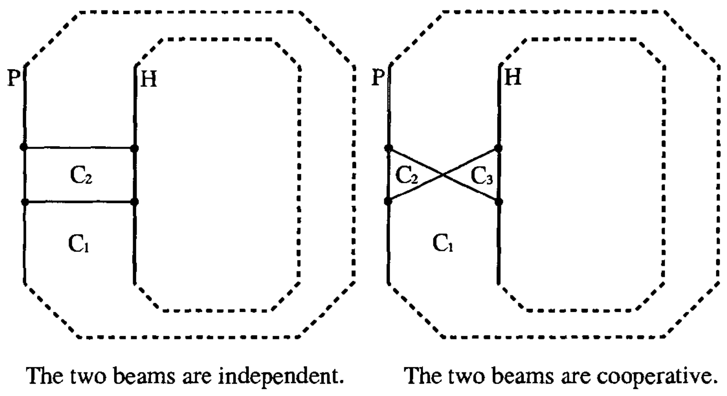

P
H
C2
C1
The two beams are independent.
P
H
C2
C3
C1
The two beams are cooperative.

Figure 5.9: The contamination classes.

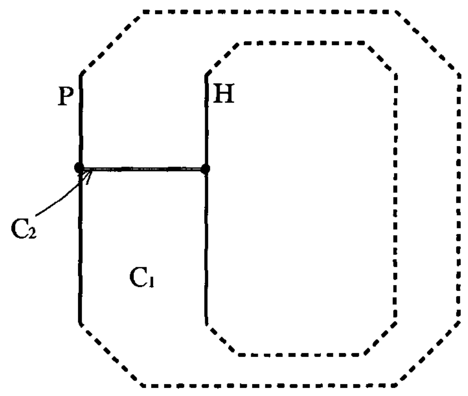


Figure 5.10: A special case where the beams $\overline{r_1q_1}$ and $\overline{r_2q_2}$ overlap.

Throughout the search, the beams of the searchers' flashlights $\overline{r_1q_1}$ and $\overline{r_2q_2}$ partition the polygon P into at most three connected components. We show the possible partitions in Figure 5.9. In these partitions, the beams from the flashlights of the two searchers could be either *independent* or *cooperative* [99]. Figure 5.9 shows an example for independent and cooperative beams. Note that if the two beams touch each other only at one of their end points, they are considered independent. We also show in Figure 5.10 a special case where the beams $\overline{r_1q_1}$ and $\overline{r_2q_2}$ overlap with each other. In order to be consistent with the situations shown in Figure 5.9, we still say that there are two components but the area of C_2 is zero.

Each component is characterized by a *contamination status.* We label each component by "0" (being cleared) or "1" (being contaminated). We use a binary string called the *contamination string*, denoted as C, to represent the current contamination statuses. Since there can be always two or three components, we have $C \in \{0,1\}^2 \cup \{0,1\}^3$.

Combining the visible configurations representing the current states of the two searchers and the contamination string representing the current contamination status, we define a triple (S_1, S_2, C) to be a *search state*, where $S_i = \langle r_i, q_i \rangle$ $(i = 1, 2)$ are visible configurations and C is a contamination string. (See more below on this.) In a 1-hole polygon, the set of all the search states is called its *search space.*

We define a search state as a *starting state* if its contamination string is "10" and S_1 and S_2 overlap. (Recall the special contamination shown in Figure 5.10.) Similarly, a *goal state* is defined as the one in which the contamination string contains "00" and S_1 and S_2 overlap.

Intuitively speaking, the search space contains all of the possible states of the searchers and the contamination status in each state. These two pieces of information are encoded in a search state. Now in order to check whether a 1-hole polygon is searchable or not, we need to find a continuous sequence of search states, which starts in a starting state and ends in a goal state, in this space. If no such sequence exists, the polygon is not searchable.

Let $S_i : [0, 1] \to \partial P \times \partial H$ $(i = 1, 2)$, such that $S_i(t) = (r_i(t), q_i(t))$ encodes the position of searcher i at time t. We will discuss below which one of $r_i(t)$ and $q_i(t)$ represents the standing position of searcher i and which one represents the beam head of her flashlight. Let $C(t) : [0, 1] \to \{0,1\}^2 \cup \{0,1\}^3$ be the contamination string at time t.

A *search schedule* starts in a starting state and is composed of the triples $(S_1(t), S_2(t), C(t))$ over time $t \in [0, 1]$. Each triple $(S_1(t), S_2(t), C(t))$ encodes the states of the searchers and the contamination string at time t.

When both searchers move on ∂H, at time t searcher i is standing at $q_i(t)$ and the beam head of her flashlight is at $r_i(t)$. The standing position of the searcher i must be continuous over time while the beam head of her flashlight must be piecewise continuous. Recall the recontaminations for a 1-searcher as discussed in Sections 2.5 and 4.3.

When both searchers move on ∂P, at time t searcher i is standing at $r_i(t)$ while the beam head of her flashlight is at $q_i(t)$. The standing position of the searcher i must be continuous over time while the beam head of her flashlight can be piecewise continuous due to its jumps over reflex vertices on ∂H.

When searcher 1 moves on ∂P and searcher 2 moves on ∂H, at time t, $r_1(t)$ and $q_2(t)$ are the standing positions of searchers 1 and 2, respectively, while $q_1(t)$ and $r_2(t)$ are the beam heads of their flashlights, respectively. For a searcher, her standing position must be continuous on the respective boundary while the beam head of her flashlight can be piecewise continuous, due to its jumps over reflex vertices on the opposite boundary.

If at time 1, a search schedule is in a goal state, we call it a *legal schedule.* The proposition below follows directly from the definitions of search states, search space and legal schedule.

Proposition 5.3.1 *A 1-hole polygon is searchable by two 1-searchers if and only if a legal schedule exists in its corresponding search space.*

5.3.1 Moves of searchers

As the two searchers move on the boundaries of a 1-hole polygon, the contamination bits change and the beams of the flashlights from the searchers may change from independent to cooperative and vice versa. In order to delineate these changes, we introduce *moves* (defined below) that comprise a legal schedule.

Observe that the functions $(S_1(t), S_2(t), C(t))$ change over time. For most of the time, $C(t)$ stays constant. There are changes to $C(t)$ when the relative order of r_1, q_1, r_2, and q_2 changes, representing that the beams change from independent to cooperative or vice and versa, perhaps due to a beam head jump over reflex vertices. However, these changes only happen at discrete moments of time. We identify different types of changes to $C(t)$ and accordingly introduce *moves.*

Type 1 moves: No beam head jumps and no relative order changes through the move

In this type of move, the two beams from the flashlights keep the relative order, i.e., at the beginning of the move, if the two beams are independent (cooperative, resp.), at the end the move, the beams are still independent (cooperative, resp.), and because of this, the contamination string does not change.

Type 2 moves: No beam head jumps but the beam's relative order changes through the move

In this type of move, the beams change from independent to cooperative or vis versa, and accordingly the information in $C(t)$ changes.

Type 3 moves: Beam head jumps

This type of move corresponds to the situation where the beam head of the flashlight from a searcher makes a jump while the state of the other searcher is stationary. The jump results in a change to the contamination string (possibly causing recontaminations) and a possible simultaneous change to the relative order of the beams, i.e., they may change from independent to cooperative or vice versa.

More precisely, for a given schedule, let the time moments $t_0 = 0 < t_1 < t_2 < \cdots < t_k = 1$ be where changes may occur in $C(t)$. Type 2 and 3 moves convert a search schedule into a sequence of search states s_1, s_2, $\cdots$, s_k. Within such a state only Type 1 moves take place, while at t_i, the changes correspond to Type 2 or 3 moves. With respect to the following conditions, we decompose a move into individual parts.

With a Type 2 or 3 move, the two beams of the flashlights could change from independent to cooperative, as shown in Figure 5.9, or vice versa. There could be at most one beam head jump. To be more specific, a move is represented as $m_s m_j m_e$, where (1) $m_s = C$ (being cooperative) if at the beginning of a move, the two beams are cooperative or $m_s = I$ (being independent) if at the beginning of the move, the beams are independent; (2) $m_j = 1$, if there is a beam head jump during the move or $m_j = 0$, otherwise; and (3) $m_e = C$, if at the end of the move the beams are cooperative or $m_e = I$, if at the end of the move the beams are independent. Note that a searcher might move her position and rotate her beam head simultaneously, as long as it is within a valid move. A Type 2 or 3 move also possibly causes the contamination string to be recomputed.

We need to consider moves for the three different search settings. When both searchers move on ∂H, we show in Figure 5.11 the possible transitions from one search state to another. In the figure, when recomputing the contamination string, the "or" operation is the standard boolean operation, i.e., when both operands are "0"s, the result is "0"; otherwise, the result is "1".

The situation where both searchers move on ∂P can be handled similarly. We show the corresponding moves in Figure 5.12.

However, the situation where one searcher moves on ∂P and the other moves on ∂H is different. We show the possible transitions in Figures 5.13 and 5.14.

Note that there is an important difference here from the results we reported in Chapter 4, in which we require that at any time, the 1-searcher maintain the so-called left invariance [67, 68] in the polygon search problem by a 1-searcher. The left invariance guarantees that at any time, the beam from the flashlight separates the cleared portion and contaminated portion of the polygon, even though one of them might have an area of zero.

However, in the 1-hole polygon search problem by two 1-searchers, from the discussions above, the contamination is encoded in the contamination string. The changes of the contamination string will follow the formulae shown in the above figures. If the contamination string contains all "0"s, then the 1-hole polygon has been searched successfully by the two 1-searchers.

5.4 Equivalence classes of search states

The search space, as defined above, of a 1-hole polygon contains all of the possible states of the searchers and the corresponding contamination status. However, this space is infinite and it is hard to find a representation that makes it feasible to seek a legal schedule.

5.4.1 Critical points

We observe that the major difficulties arising when searching a polygon are due to the reflex vertices. On the other hand, a consecutive sequence of non-reflex vertices will not pose any problems since we can easily move the searchers over or direct their flash lights at these vertices one by one. Thus, as in the previous chapters, we can focus our attention on the reflex vertices of a 1-hole polygon.

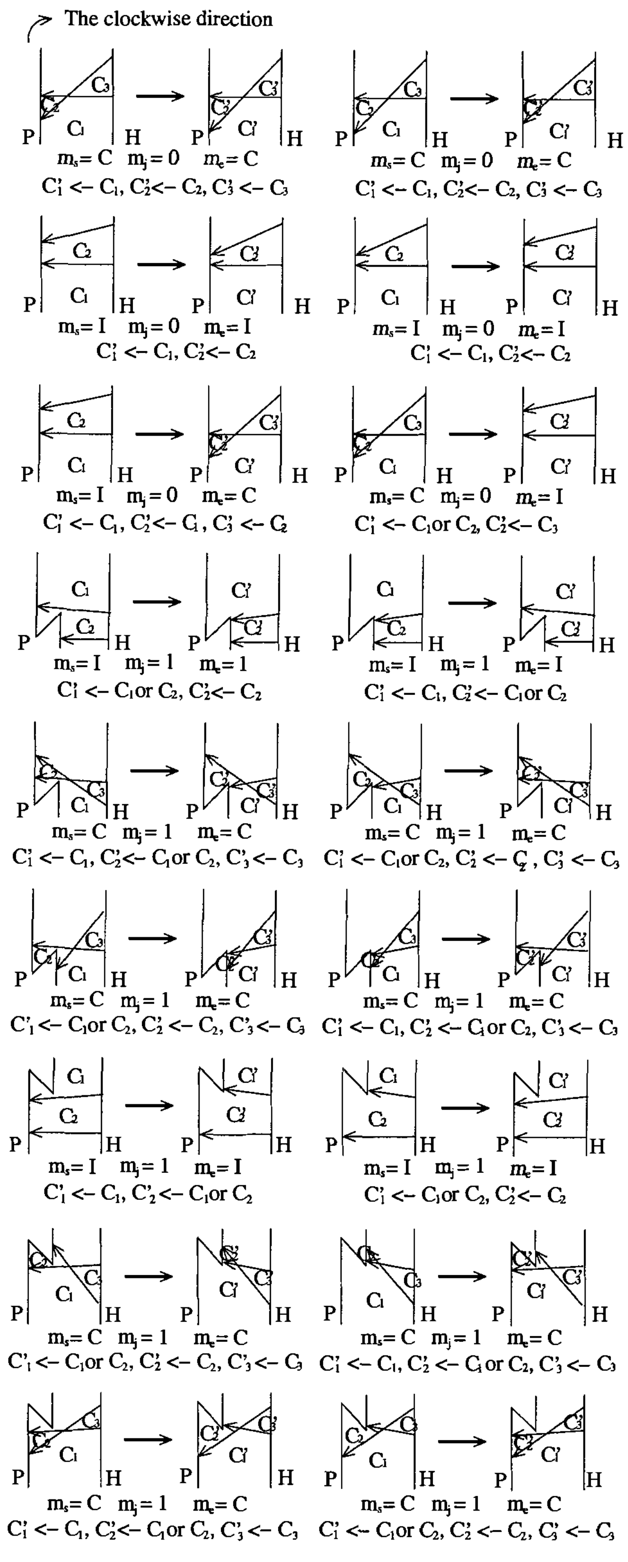

The clockwise direction
m_s= C m_j= 0 m_e= C
C'_1 <- C_1, C'_2 <- C_2, C'_3 <- C_3
m_s= C m_j= 0 m_e= C
C'_1 <- C_1, C'_2 <- C_2, C'_3 <- C_3
m_s= I m_j= 0 m_e= I
C'_1 <- C_1, C'_2 <- C_2
m_s= I m_j= 0 m_e= I
C'_1 <- C_1, C'_2 <- C_2
m_s= I m_j= 0 m_e= C
C'_1 <- C_1, C'_2 <- C_1, C'_3 <- C_2
m_s= C m_j= 0 m_e= I
C'_1 <- C_1 or C_2, C'_2 <- C_3
m_s= I m_j= 1 m_e= I
C'_1 <- C_1 or C_2, C'_2 <- C_2
m_s= I m_j= 1 m_e= I
C'_1 <- C_1, C'_2 <- C_1 or C_2
m_s= C m_j= 1 m_e= C
C'_1 <- C_1, C'_2 <- C_1 or C_2, C'_3 <- C_3
m_s= C m_j= 1 m_e= C
C'_1 <- C_1 or C_2, C'_2 <- C_2, C'_3 <- C_3
m_s= C m_j= 1 m_e= C
C'_1 <- C_1 or C_2, C'_2 <- C_2, C'_3 <- C_3
m_s= C m_j= 1 m_e= C
C'_1 <- C_1, C'_2 <- C_1 or C_2, C'_3 <- C_3
m_s= I m_j= 1 m_e= I
C'_1 <- C_1, C'_2 <- C_1 or C_2
m_s= I m_j= 1 m_e= I
C'_1 <- C_1 or C_2, C'_2 <- C_2
m_s= C m_j= 1 m_e= C
C'_1 <- C_1 or C_2, C'_2 <- C_2, C'_3 <- C_3
m_s= C m_j= 1 m_e= C
C'_1 <- C_1, C'_2 <- C_1 or C_2, C'_3 <- C_3
m_s= C m_j= 1 m_e= C
C'_1 <- C_1, C'_2 <- C_1 or C_2, C'_3 <- C_3
m_s= C m_j= 1 m_e= C
C'_1 <- C_1 or C_2, C'_2 <- C_2, C'_3 <- C_3

Figure 5.11: The moves when both searchers are on ∂H.

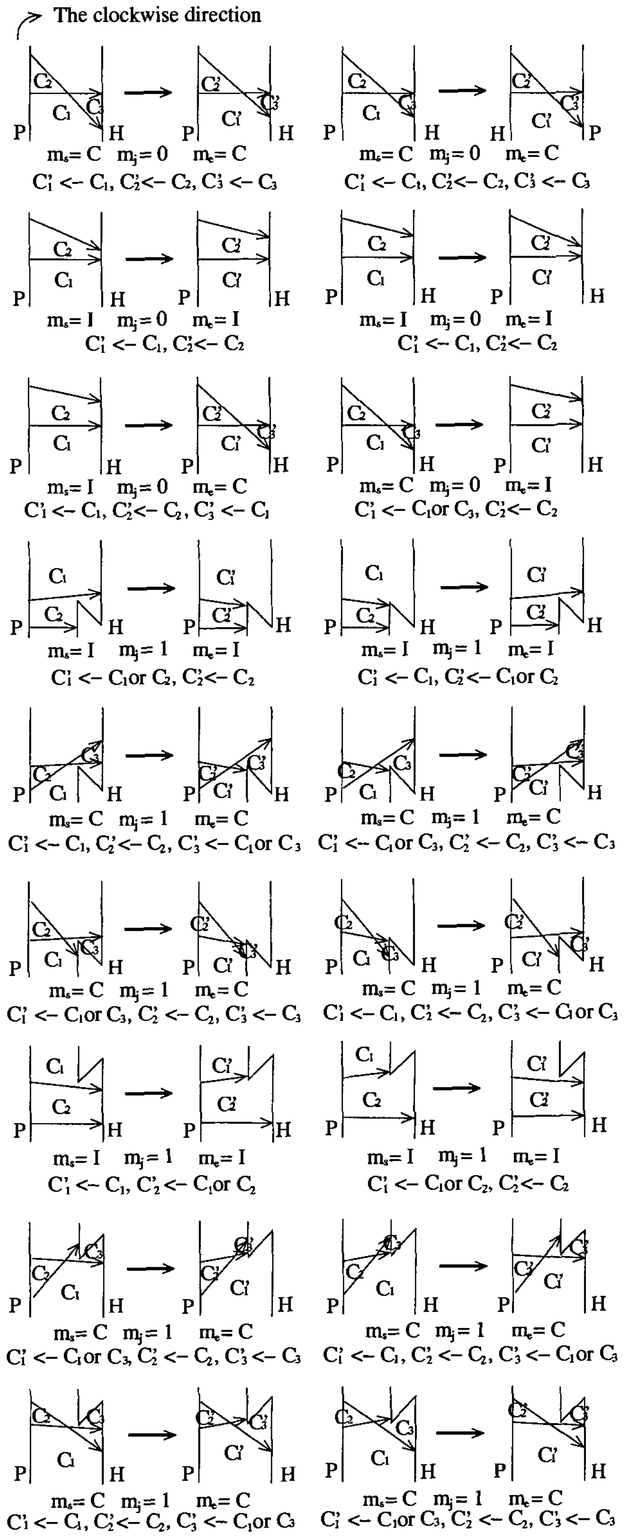


Figure 5.12: The moves when both searchers are on ∂P.

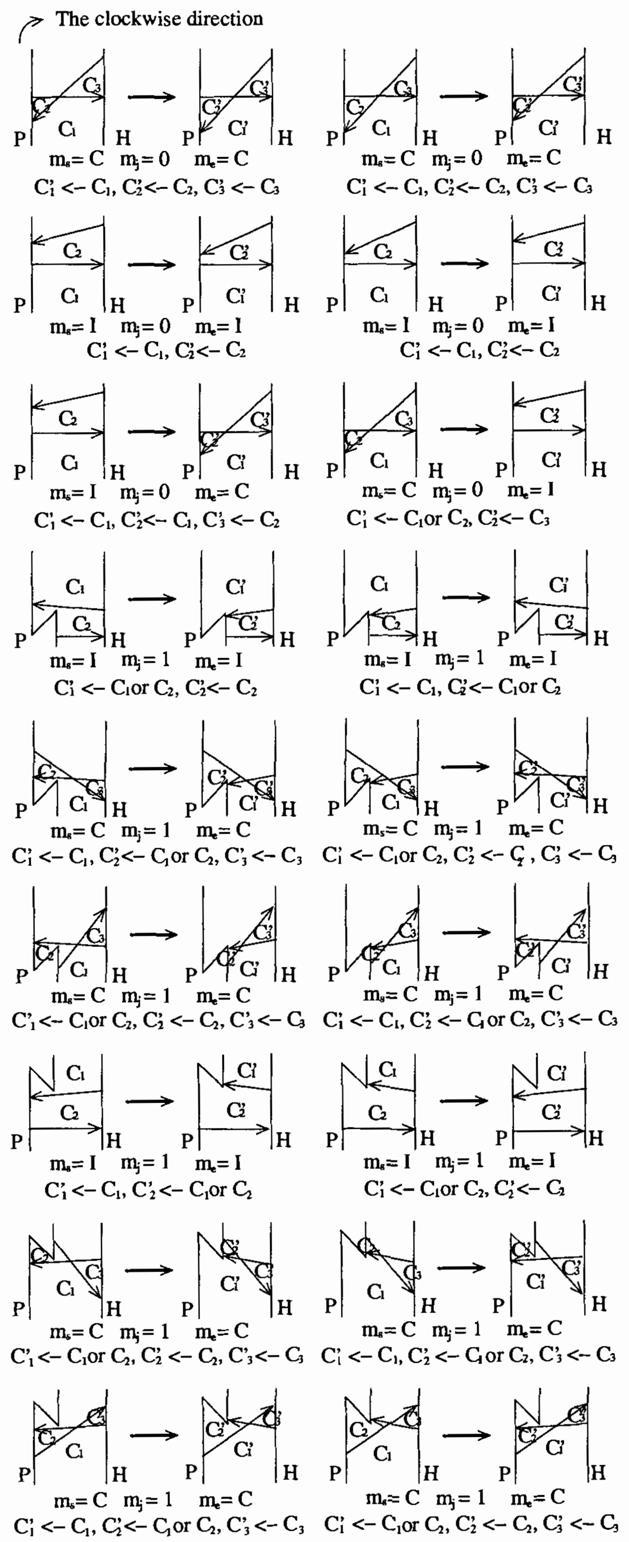

The clockwise direction
m_s = C m_j = 0 m_e = C
C'1 <- C1, C'2 <- C2, C'3 <- C3
m_s = C m_j = 0 m_e = C
C'1 <- C1, C'2 <- C2, C'3 <- C3
m_s = I m_j = 0 m_e = I
C'1 <- C1, C'2 <- C2
m_s = I m_j = 0 m_e = I
C'1 <- C1, C'2 <- C2
m_s = I m_j = 0 m_e = C
C'1 <- C1, C'2 <- C1, C'3 <- C2
m_s = C m_j = 0 m_e = I
C'1 <- C1 or C2, C'2 <- C3
m_s = I m_j = 1 m_e = I
C'1 <- C1 or C2, C'2 <- C2
m_s = I m_j = 1 m_e = I
C'1 <- C1, C'2 <- C1 or C2
m_s = C m_j = 1 m_e = C
C'1 <- C1, C'2 <- C1 or C2, C'3 <- C3
m_s = C m_j = 1 m_e = C
C'1 <- C1 or C2, C'2 <- C2, C'3 <- C3
m_s = C m_j = 1 m_e = C
C'1 <- C1 or C2, C'2 <- C2, C'3 <- C3
m_s = C m_j = 1 m_e = C
C'1 <- C1, C'2 <- C1 or C2, C'3 <- C3
m_s = I m_j = 1 m_e = I
C'1 <- C1, C'2 <- C1 or C2
m_s = I m_j = 1 m_e = I
C'1 <- C1 or C2, C'2 <- C2
m_s = C m_j = 1 m_e = C
C'1 <- C1 or C2, C'2 <- C2, C'3 <- C3
m_s = C m_j = 1 m_e = C
C'1 <- C1, C'2 <- C1 or C2, C'3 <- C3
m_s = C m_j = 1 m_e = C
C'1 <- C1, C'2 <- C1 or C2, C'3 <- C3
m_s = C m_j = 1 m_e = C
C'1 <- C1 or C2, C'2 <- C2, C'3 <- C3

Figure 5.13: The moves when one searcher moves on ∂P while the other moves on ∂H.

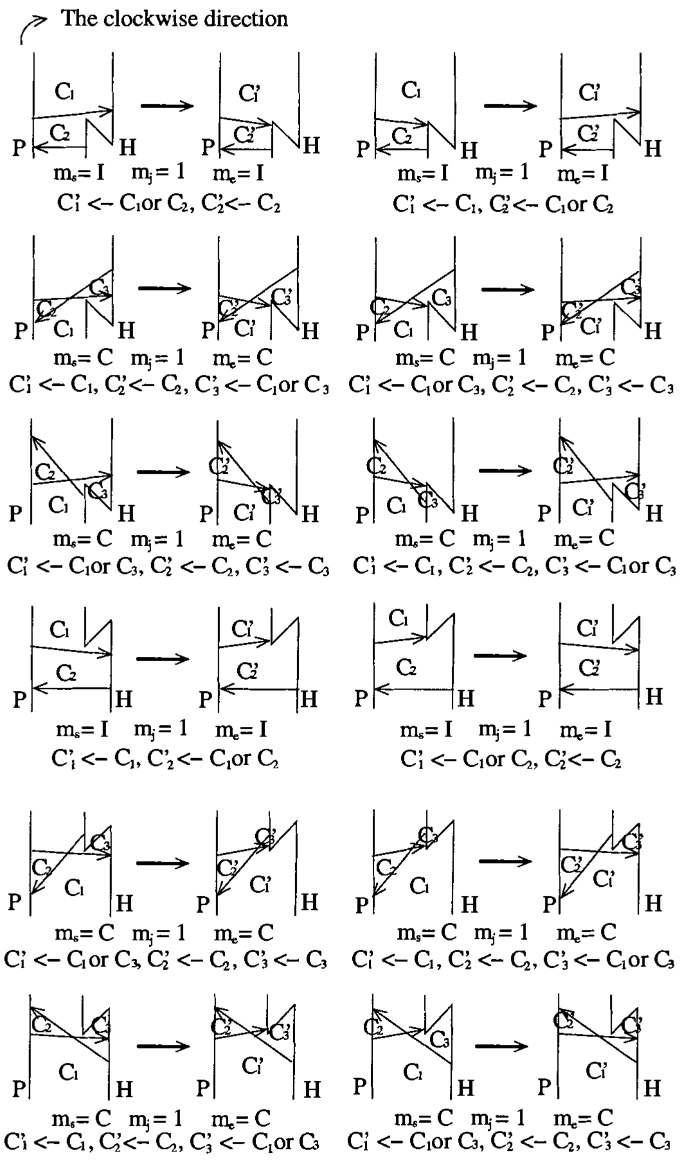

The clockwise direction
m_s= I m_j= 1 m_e= I
C'₁ <- C₁or C₂, C'₂<- C₂
m_s= I m_j= 1 m_e= I
C'₁ <- C₁, C'₂<- C₁or C₂
m_s= C m_j= 1 m_e= C
C'₁ <- C₁, C'₂<- C₂, C'₃ <- C₁or C₃
m_s= C m_j= 1 m_e= C
C'₁ <- C₁or C₃, C'₂ <- C₂, C'₃ <- C₃
m_s= C m_j= 1 m_e= C
C'₁ <- C₁or C₃, C'₂ <- C₂, C'₃ <- C₃
m_s= C m_j= 1 m_e= C
C'₁ <- C₁, C'₂ <- C₂, C'₃ <- C₁or C₃
m_s= I m_j= 1 m_e= I
C'₁ <- C₁, C'₂<- C₁or C₂
m_s= I m_j= 1 m_e= I
C'₁ <- C₁or C₂, C'₂<- C₂
m_s= C m_j= 1 m_e= C
C'₁ <- C₁or C₃, C'₂ <- C₂, C'₃ <- C₃
m_s= C m_j= 1 m_e= C
C'₁ <- C₁, C'₂ <- C₂, C'₃ <- C₁or C₃
m_s= C m_j= 1 m_e= C
C'₁ <- C₁, C'₂<- C₂, C'₃ <- C₁or C₃
m_s= C m_j= 1 m_e= C
C'₁ <- C₁or C₃, C'₂ <- C₂, C'₃ <- C₃

Figure 5.14: The moves when one searcher moves on ∂P while the other moves on ∂H (cont'd).

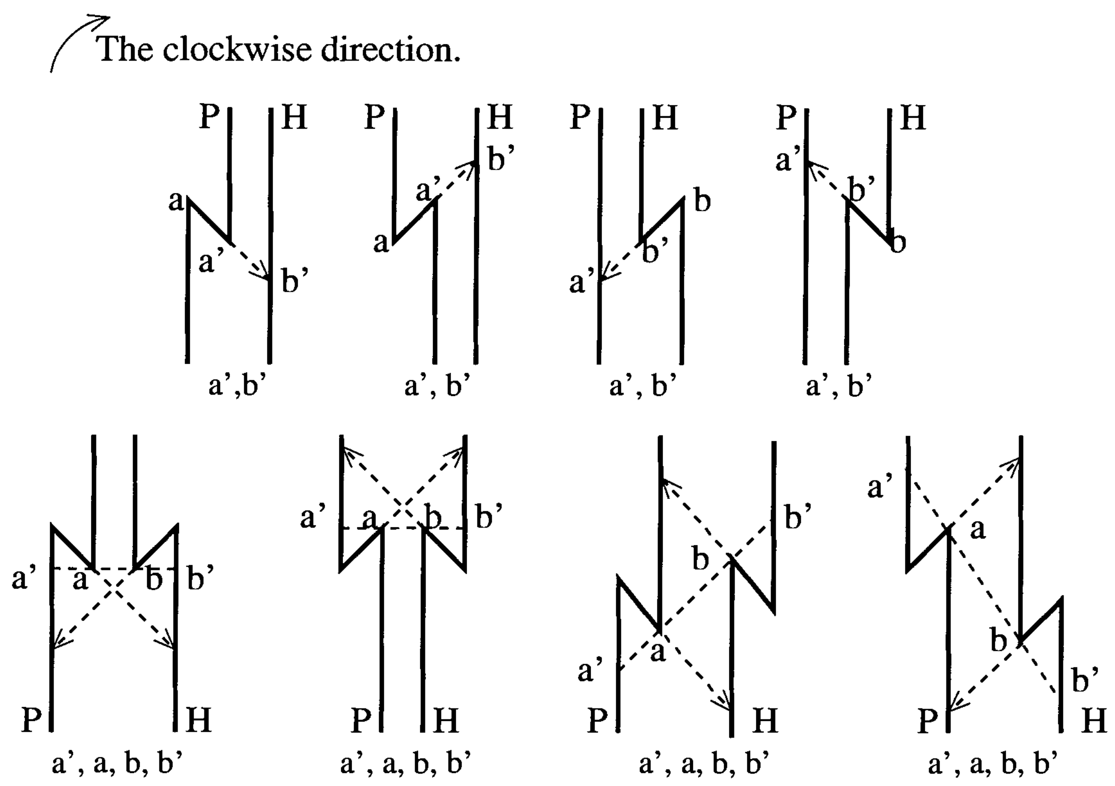


Figure 5.15: Critical points.

We consider the patterns shown in Figure 5.15. We define a *critical point* to be a reflex vertex, a forward ray end point, a backward ray end point, or an end point of a *bitangent*. One bitangent introduces two end points on the boundaries of the 1-hole polygon, as shown in Figure 5.15.

From these critical points, we define a *section* to be either a single reflex vertex or composed of the part of a polygon boundary (i.e., ∂P and ∂H) that lies between two adjacent critical points. All sections are disjoint. Therefore, if a section that is not a reflex vertex is bounded by a reflex vertex, then it is open at that end. In general, a section might be closed and open at both ends. Note that if a critical point that is not a reflex vertex is at the interface of two sections, we select arbitrarily one section to be closed at the critical point while letting the other section to be open.

For consistency, we call the sections on ∂P R_0, R_1, $\cdots$, R_{n_1-1}, while the sections on ∂H are represented by Q_0, Q_1, $\cdots$, Q_{n_2-1}, where n_1 and n_2 (of order $O(n)$) are the number of sections on ∂P and ∂H, respectively.

In order to make the following discussions easier, we identify a section by a number. In particular, we designate the sections on ∂P as 0, 1, $\cdots$, i, $\cdots$, $n_1 - 1$ and the sections on ∂H as 0, 1, $\cdots$, j, $\cdots$, $n_2 - 1$.

5.4.2 Equivalence classes

Let R and Q be sections on ∂P and ∂H, respectively. We call (R, Q) a *section pair.* Consider tuples $D_v = (\langle r_1, q_1\rangle, \langle r_2, q_2\rangle)$ and $D_v{}' = (\langle r_1{}', q_1{}'\rangle, \langle r_2{}', q_2{}'\rangle)$, where $\langle r_1, q_1\rangle$, $\langle r_2, q_2\rangle$, $\langle r_1{}', q_1{}'\rangle$ and $\langle r_2{}', q_2{}'\rangle$ are visible configurations. We say that D_v is *similar* to $D_v{}'$ if there exist two section pairs (R_{i_1}, Q_{j_1}) and (R_{i_2}, R_{j_2}) such that $r_1, r_1{}' \in R_{i_1}$, $r_2, r_2{}' \in R_{i_2}$, $q_1, q_1{}' \in Q_{j_1}$ and $q_2, q_2{}' \in Q_{j_2}$. This relationship can be extended to a relation on search states. Let C be a contamination string. Let D and D' be two search states such that D is a concatenation of D_v and C and D' is a concatenation of $D_v{}'$ and C. We say that D and D' are *similar* if D_v and $D_v{}'$ are similar. This "similar" relation partitions the search space into a set of equivalence classes of the form $((R_{i_1}, Q_{j_2}), (R_{j_1}, Q_{j_2}), C)$.

Note that a section pair (R_i, Q_j) can be simply represented by (i, j), and there are 4 section pairs adjacent to it, i.e., $(i+1 \pmod{n_1}, j)$, $(i-1 \pmod{n_1}, j)$, $(i, j+1 \pmod{n_2})$, and $(i, j-1 \pmod{n_2})$.

We show the possible equivalence classes in Figure 5.16. In the figure, R, R', R'', Q, Q' and Q'' are sections while $C'C''$ and $C'C''C'''$ are the contamination strings for a particular class. One special situation which is not shown in the figure is where a section is a reflex vertex as a degenerate case. Note that for the four equivalence classes in the second row in the figure, Type 2 and Type 3 moves are possible.

In order to study the properties of equivalence classes over the search space, we superimpose a grid over the VD of a 1-hole polygon, defined by the critical points. In this grid, a section pair (R, Q) is represented by a rectangle defined by the grid element corresponding to R on ∂P and to Q on ∂H, respectively, and infinitely many copies of this grid obtained by shifting by $k_1|\partial P|$ horizontally and $k_2|\partial P|$ vertically, where k_1 and k_2 are integers. In what follows, we normally concentrate on one copy corresponding to the situation when $k_1 = k_2 = 0$.

We call such a rectangle $Box(R, Q)$ in the following. If $Box(R, Q)$ intersects the white area of the VD, then we call the intersection a *core cell* of $Box(R, Q)$ and denote it as $CC(R, Q)$. $Box(R, Q)$ and $CC(R, Q)$ are similar to the *vertically and horizontally convex box* (VHC) and the VHC *core*, respectively, introduced by Simov *et al.* [100] and LaValle *et*

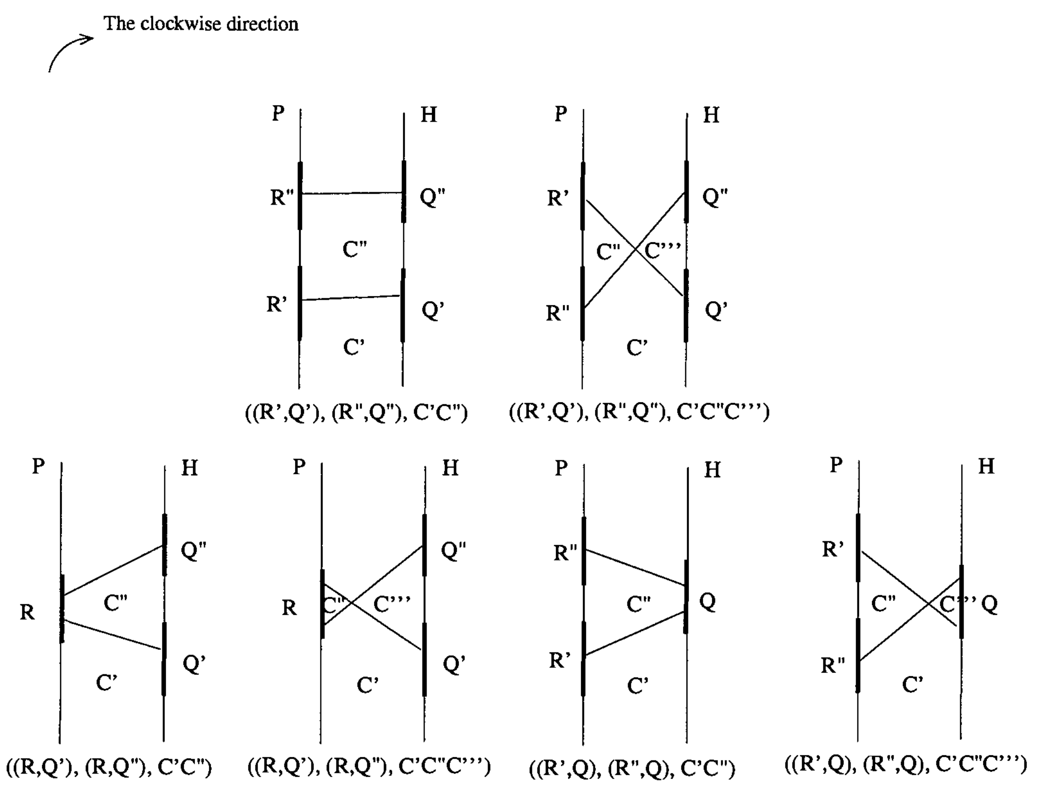


Figure 5.16: Different equivalence classes induced by the similarity among section pairs.

al. [67, 68].

Figure 5.17 shows $Box(R, Q)$ and $CC(R, Q)$ for a section pair (R, Q). For the polygon and its VD shown in Figure 5.8, we show its critical points in Figure 5.18 (a). Consider the critical points a'_2 and a'_3 on ∂P and the critical points b'_2 and b'_3 on ∂H. Those critical points are due to the ray end points and the bitangent from reflex vertices b'_1 and a'_1.

Let $R = \partial P_{cw}(a'_2, a'_3)$ and $Q = \partial H_{cw}(b'_2, b'_3)$. We show section pair (R, Q) in Figure 5.18 (a) and its rectangle box and core cell in Figure 5.18 (b), respectively.

A $CC(R, Q)$ has a nice property. It is called the *vertical and horizontal convexity* [100], which tells us that for every $r \in R$, the set $\{q | (r, q) \in CC(R, Q)\}$ is a single connected non-empty interval and for every $q \in Q$, the set $\{r | (r, q) \in CC(R, Q)\}$ is a single non-empty interval. In the following we prove this property using our notation.

Lemma 5.4.1 *Consider the grid over the VD of a 1-hole polygon. Let $Box(R, Q)$ be a grid element. If $CC(R, Q)$ is not empty, then it has the vertical and horizontal convexity.*

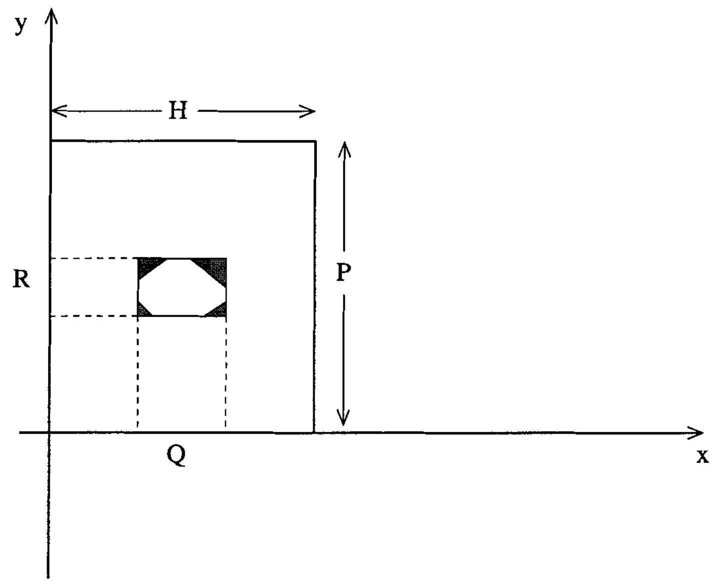


Figure 5.17: $Box(R, Q)$ and its core cell.

Proof If at least one of R and Q is a reflex vertex in the degenerate case, then the $CC(R, Q)$ degenerates to a point or a vertical or horizontal line segment. The lemma follows trivially.

Now suppose that neither R nor Q is a reflex vertex. No point within R or Q can be a critical point (and a reflex vertex). See Figure 5.15.

We consider the horizontal convexity here. The vertical convexity is dealt with similarly. Let $R = \partial P_{cw}(r_1, r_2)$, where both r_1 and r_2 are critical points, and let $Q = \partial H_{cw}(q_1, q_2)$, where both q_1 and q_2 are critical points. Recall that in Chapter 2, the gray areas in VOD (VD, in this context) are due to reflex vertices. In particular, for any gray area due to a reflex vertex, there is a curved side. See Figure 2.4. This curved side is convex toward the interior of the gray area, as discussed in Section 2.3.

Consider $Box(R, Q)$ for a section pair (R, Q). As discussed above, no reflex vertex can be within R or Q. This means that the rectangle representing $Box(R, Q)$ can only overlap with a gray area on its curved side, if such an overlap exists. Since the curved side is convex toward the interior of the gray area, the part of the gray area within the rectangle must be concave toward the interior of $CC(R, Q)$.

The rectangle may overlap several gray areas. But each overlap results in a concave gray area, as the one discussed above on the shape of $CC(R, Q)$. The lemma then follows. Note that the gray area in $CC(R, Q)$ may come from the same contiguous gray area. ▮

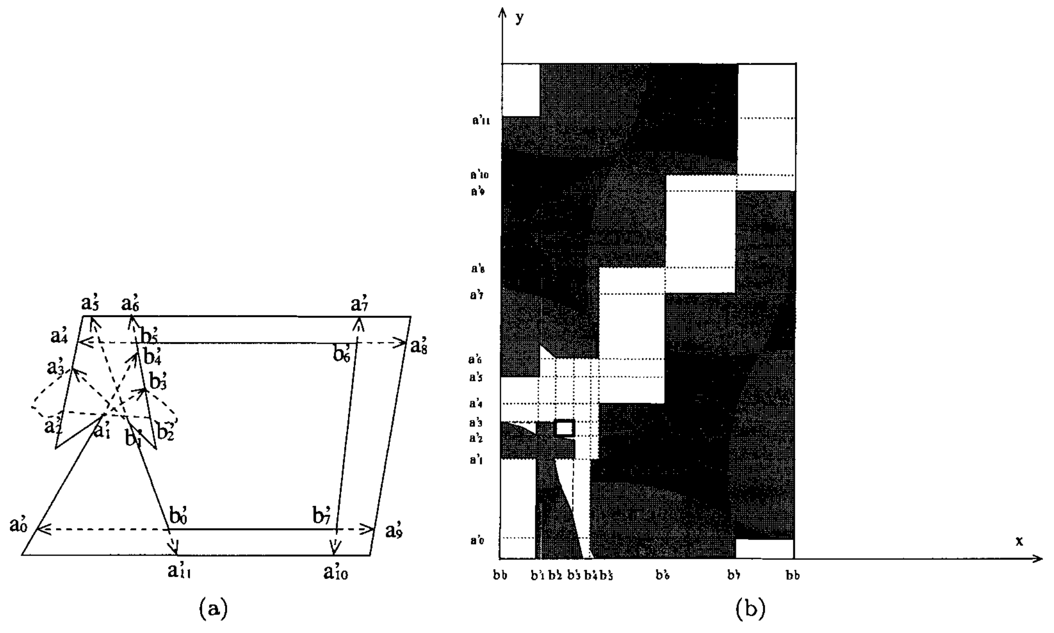


Figure 5.18: A section pair and its rectangle box and core cell.

Now it is easy to see that if there are gray areas and white area inside $Box(R, Q)$, then the gray areas only occupy some of the four corners of the rectangle. Thus $CC(R, Q)$ always consists of a single white area inside $Box(R, Q)$.

Consider $Box(R_1, Q_1)$ and $Box(R_2, Q_2)$. We connect a white point in $CC(R_1, Q_1)$ to a white point in $CC(R_2, Q_2)$. If the slope of the line segment connecting those two points is positive, then it is easy to see that at this moment the two beams of the flashlights are independent. On the other hand, if the slope is negative, the two beams are cooperative. This situation is shown in Figure 5.19.

If the line segment is vertical or horizontal, then we have the situation where R_1 is equal to R_2 or Q_1 is equal to Q_2. In this case, the two beams is either independent or cooperative. Another special case is where one or both the sections in a section pair can be a reflex vertex. In this case, we still say that there is a corresponding rectangle box but its area is zero.

It is easy to see from the definition of a core cell that if D_1 and D_2 are two search states contained in the same equivalence class of the search space, then a Type 1 move suffices to

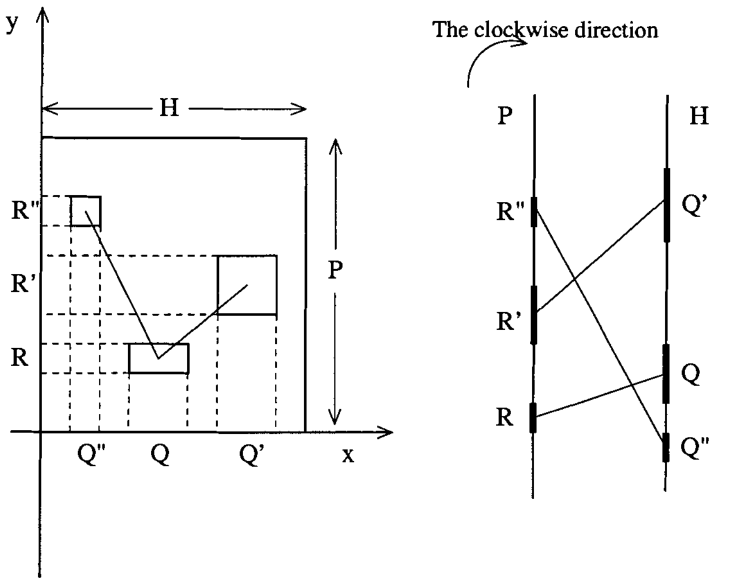


Figure 5.19: The relationship between two core cells.

change from D_1 to D_2, or from D_2 to D_1. On the other hand, Type 2 and Type 3 moves can change a search state in an equivalence class to a search state in another equivalence class.

5.5 Search graph of the visibility space

We define a directed *search graph* $G_s = (V_s, E_s)$ to capture the equivalence classes of the search states. Each equivalence class in the search space is represented by a vertex in V_s. The set of edges, E_s, consists of all the pairs (v', v'') such that there is a move from some search state in v' to some search state in v''.

A vertex $v \in V_s$ is called a *starting vertex* if the equivalence class represented by v contains a starting search state while a $v' \in V_s$ is called a *goal vertex* if the equivalence class represented by v' contains a goal search state. A *legal path* is a path in G_s from a starting vertex to a goal vertex.

Lemma 5.5.1 *For a 1-hole polygon, there exists a legal schedule in the search space if and only if there exists a legal path in G_s.*

Proof Suppose that there is a schedule σ in the search space of the polygon from a starting search state to a goal state, such that σ consists of moves defined above.

The schedule can be partitioned into subschedules $\sigma_0, \sigma_1, \cdots, \sigma_k$, where each σ_i is a sequence of search states within the same equivalence class v_i $(0 \leq i \leq k)$. Since σ_0 begins from a starting search state, v_0 is a starting vertex in G_s. For the intermediate subschedules, the transition from σ_{i-1} to σ_i corresponds to a move, which means that $(v_{i-1}, v_i) \in E_s$. Now connecting the vertices v_i using the edges between them and considering that σ_k ends in a goal search state (which means that v_k is a goal vertex in G_s), we have a legal path in G_s.

Conversely, suppose that there exists a legal path $v_0, v_1, \cdots, v_k$, where v_0 is a starting vertex while v_k is a goal vertex. We consider how to convert this path into a legal schedule in the search space. Since vertex v_0 is a starting vertex, we can find a starting search state s'_0 in the equivalence class represented by it while since vertex v_k is a goal vertex, we can find a goal search state s''_k in the equivalence class represented by it as well. Furthermore, since (v_{i-1}, v_i), where $i = 1,\cdots,k$, belongs to E_s, it follows that there exists a move (could be of any move type) which changes search state s''_{i-1} in v_{i-1} to search state s'_i in v_i. Within the same equivalence class represented by v_i, where $i = 0, \cdots, k$, we select a sequence of search states that starts at s'_i and ends at s''_i (Note that we have already selected s'_0 for v_0 and s''_k for v_k as above, respectively.), and use a Type 1 move for the two searchers to move from each search state to the next in this sequence. We denote this sequence as σ'_i, and construct a schedule by concatenating $\sigma'_0, \sigma'_1, \cdots, \sigma'_k$ together. The schedule starts from a starting search state and ends at a goal search state. Therefore, we have obtained a legal schedule in the search space. ▮

The above proof is similar to the one proposed by Simov *et al.* [100].

Suppose that the two section pairs and contamination string in the equivalence class represented by a goal vertex v_k are (R_{i_1}, Q_{j_1}), (R_{i_2}, Q_{j_2}) and C, then C contains all "0"s and $CC(R_{i_1}, Q_{j_1})$ and $CC(R_{i_2}, Q_{j_2})$ are separated by the distance $|\partial H|$ horizontally and by the distance $|\partial P|$ vertically in the VD of the polygon, reflecting the fact the two beams have met after the searchers have gone around the 1-hole polygon in the generally opposite direction. Note that for $0 \leq i \leq n_1 - 1$, R_i appears infinitely many times with period $|\partial P|$. Similarly, for $0 \leq j \leq n_2 - 1$, Q_j appears infinitely many times with period $|\partial H|$. Though there are an infinite number of sections (due to the circularity of ∂P and ∂H), the number of different sections is finite. As commented earlier, we only consider the area of VD that is in the range $[0, |\partial H|]$ on the horizontal axis and in the range $[0, |\partial P|]$ on the vertical axis.

Thus, each vertex in G_s can be described by $((i_1, j_1), (i_2, j_2), C)$, where $0 \le i_1, i_2 \le n_1 - 1$ and $0 \le j_1, j_2 \le n_2 - 1$.

5.6 Time complexity for finding a legal schedule

We construct graph G_s which represents the equivalence classes of the search states and perform a breadth-first search (BFS) on it to find a legal path. If such a path exists, then the corresponding 1-hole polygon is searchable and the path can be converted into a search schedule for the two searchers [99, 100].

To analyze the time complexity, suppose that ∂P and ∂H together contain n vertices. There can be at most $O(n)$ sections. We need at most $O(n \log n)$ time to calculate all the critical points due to the reflex vertices and the ray end points from them [12]. As for the critical points due to bitangents, since there are n vertices, the total time to determine them is $O(n^2)$, i.e., we need to check pairwise reflex vertices on ∂P and ∂H.

Note that, as discussed before, a section pair (R_i, Q_j) can be simply represented by (i, j), and there are 4 section pairs adjacent to it, i.e., $(i + 1 \pmod{n_1}, j)$, $(i - 1 \pmod{n_1}, j)$, $(i, j + 1 \pmod{n_2})$, and $(i, j - 1 \pmod{n_2})$.

For section pairs, we maintain a 2-dimensional array indexed by (i, j), where $i = 0, \cdots, n_1 - 1$ and $j = 0, \cdots, n_2 - 1$. Access time to this array is O(1). We can also immediately find the adjacent section pairs from the indices, as discussed above. Each element in this array points to a structure which contains the information for the ith section and jth section on ∂P and ∂H, respectively.

There can be $O(n^2)$ section pairs. For two section pairs represented by (i_1, j_1) and (i_2, j_2), we can evaluate in $O(1)$ time whether there exist a visible configuration $\langle r_1, q_1 \rangle \in CC(R_{i_1}, Q_{j_1})$ and a visible configuration $\langle r_2, q_2 \rangle \in CC(R_{i_2}, Q_{j_2})$ [100]. This is equivalent to checking whether two section pairs could be an equivalence class together with a contamination status, i.e., a vertex in G_s. Consider all the possible choices of i_1, j_1, i_2, and j_2. Since the number of this combination is at most $O(n^4)$, so the vertex set of G_s can be constructed in $O(n^4)$ time and there are $O(n^4)$ vertices in it. This vertex set can be maintained in a 4-dimensional array (constructed from the above two 2-dimensional arrays). Each element in the array points to a data structure which stores a set of contamination strings that are possible with the two section pairs. Each contamination string points to a set of neighbor vertices in G_s, determined as described below. A tuple $((i_1, j_1), (i_2, j_2), C)$ can be accessed in $O(1)$ time in this array.

Since the move of the searchers (except for the beam head jumps. See below for more on this.) must be continuous, for a given element in this 4-dimensional array, there are only a constant number of adjacent elements (and a small number of possible contamination strings in these elements) in the array, i.e., the searchers can only move from the current search state to the adjacent search state (discussed above) contained in these elements. In order to construct the edge set of G_s, we need to check for each element in the 4-dimensional array the possible moves from one of the equivalence classes in it to an equivalence class in the adjacent elements in the array. If the destination equivalence class has the same contamination string as the source equivalence class, a Type 1 move is involved. Otherwise, a Type 2 move is employed. In both cases, we need a constant time to make such a decision [100]. For instance, when a Type 2 move is involved, we only need to check whether the change of the contamination string from the source equivalence class to the destination equivalence class is consistent with the contamination status change for a Type 2 move. Clearly this can be done in constant time. Since we need to do this for a constant number of times for a given element, the total time for constructing the edge set under the Type 1 or Type 2 moves is $O(n^4)$.

Type 3 moves cause beam head jumps. So the main goal is to decide whether there is a feasible jump between a pair of source and destination search states. Let v^1 be the equivalence class represented by $((i_1^1, j_1^1), (i_2^1, j_2^1), C^1)$ and let v^2 be the one represented by $((i_1^2, j_1^2), (i_2^2, j_2^2), C^2)$. v^1 and v^2 can be identified in the 4-dimensional array in $O(1)$. Suppose that the beam head jumps on ∂H. According to [100], we first identify the *jump intervals.* Consider one section $R_{i_1}^1$ $(= R_{i_2}^2)$ on ∂P and two sections $Q_{j_1}^1$ and $Q_{j_1}^2$ on ∂H. Let $(p', p'') \in R_{i_1}^1$ be a maximal interval with the property such that for every $p \in (p', p'')$, there is a jump for the beam head from $q_1 \in Q_{j_1}^1$ to $q_2 \in Q_{j_1}^2$. (p', p'') is called a jump interval and we use it to group *equivalent jumps.* The number of jump intervals is $O(n^2)$. Given a jump interval, $R_{i_1}^1$, $Q_{j_1}^1$, $Q_{j_1}^2$ are fixed, and there are $O(n^2)$ possible elements for i_2^1 $(= i_2^2)$ and j_2^1 $(= j_2^2)$. We need to check the possible edges from these elements in the 4-dimensional array. This can be done by further identifying different areas on the VD of the polygon. The total time for constructing all the possible edges in this case is $O(n^4)$ [100]. Note that the neighboring elements may not be the adjacent ones of the given element in the 4-dimensional array. However, there are a constant number of such neighboring elements in G_s for a given element. For the detailed analysis, a reader is referred to [100].

Therefore, we can construct the edge set of $E(G_s)$ in $O(n^4)$, whose size is $O(n^4)$ as well [99, 100]. The time complexity for performing BFS on G_s is $O(|V_s| + |E_s|) = O(n^4)$.

Note that though the search setting where both searchers move on ∂P is treated similarly (in the moves) to the one where both searchers move on ∂H, the class of 1-hole polygons searchable in the former setting could be different from the one in the latter setting.

5.7 Some examples

We show some simple examples using the graph-based search approach we have presented above.

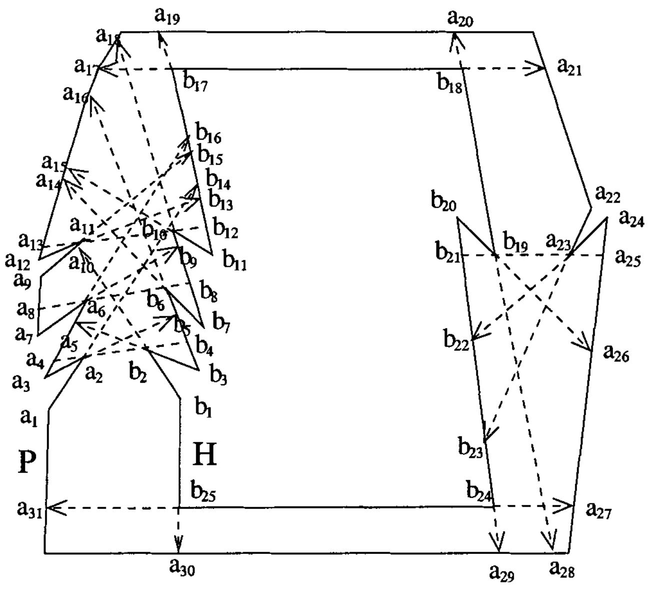


Figure 5.20: The 1-hole polygon in Figure 5.6 with relevant configurations indicated.

In order to make a simple presentation, we just use configurations in the search states in the following to describe the moves of the two searchers. The 1-hole polygon shown in Figure 5.20 is the same one as shown in Figure 5.6, with critical points indicated (we only label the relevant points). There are four deadlocks formed by the pairs of vertices a_2 and b_2, a_6 and b_6, a_{11} and b_{10}, and a_{23} and b_{19}. The first three deadlocks are located quite close to one another. We consider the situation where both searchers move on ∂H. In the following discussions, the first searcher always tries to move in the general clockwise direction while the second searcher always tries to move in the general counter-clockwise direction. Note

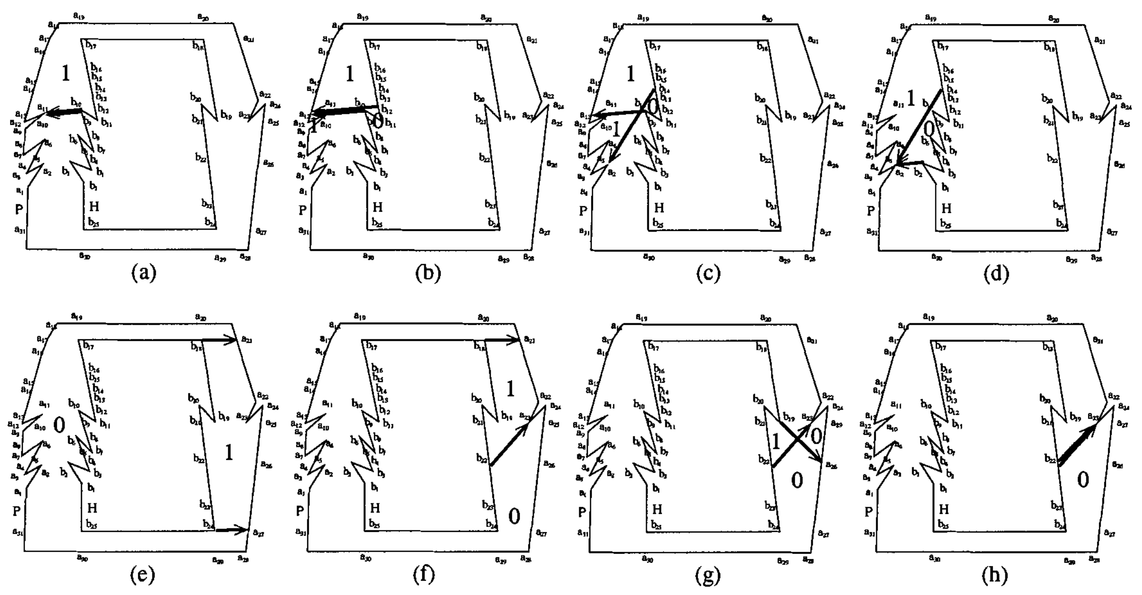


Figure 5.21: Searching the 1-hole polygon in Figure 5.6.

that sometimes it is necessary for them to move in the opposite direction from their general travel directions in order to work cooperatively.

We now show how to clear this 1-hole polygon using the moves discussed above when the two searchers move on ∂H. We use Figure 5.21 for our reference. Note that in the figure we have deleted the dashed curves in order to generate a clear illustration.

Suppose that initially the two searchers are both in configurations $\langle a_{11}, b_{10} \rangle$ and the area in between them is clear, as shown in Figure 5.10. We show this initial situation in Figure 5.21 (a). The next move for the second searcher is to configuration $\langle a_{13}, b_{10} \rangle$. We keep the first searcher stationary. We then keep the second searcher stationary and move the first searcher to configuration $\langle a_{13}, b_{10} \rangle$.

After this, the first searcher moves to configuration $\langle a_{15}, b_{10} \rangle$ and then to configuration $\langle a_{15}, b_{11} \rangle$. The first searcher continues her move to configuration $\langle a_{13}, b_{12} \rangle$. At this moment, the contamination status (represented by the contamination string in a search state) is shown in Figure 5.21 (b). The area formed by b_{10}, b_{11}, b_{12} and the beam from the first searcher is clear. Then the first searcher rotates her flashlight left to configuration $\langle a_{11}, b_{12} \rangle$. After this, the first searcher keeps moving along ∂H until she is in configuration $\langle a_{11}, b_{13} \rangle$. She next advances to the next configuration $\langle a_{11}, b_{15} \rangle$. Then she moves to the next configuration $\langle a_6, b_{16} \rangle$ and continues to configuration $\langle a_2, b_{14} \rangle$. We show the contamination status at this moment in Figure 5.21 (c).

After this, we start moving the second searcher while keeping the first one stationary. It is easy to see that the reflex vertices on ∂H between b_2 and b_{10} in the clockwise direction will not pose any difficulties to the second searcher. Eventually, by following the configurations between b_2 and b_{10}, the second searcher is able to reach configuration $\langle a_2, b_2 \rangle$. This is shown in Figure 5.21 (d). Note that at this moment the first searcher is in configuration $\langle a_2, b_{14} \rangle$ while the second searcher is in configuration $\langle a_2, b_2 \rangle$.

By following configurations $\langle a_3, b_{16} \rangle$, $\langle a_6, b_{14} \rangle$, $\langle a_9, b_{15} \rangle$, and $\langle a_{11}, b_{15} \rangle$, the first searcher can reach configuration $\langle a_{17}, b_{17} \rangle$. Eventually, the first searcher can reach configuration $\langle a_{21}, b_{18} \rangle$. Using the same rationale, the second searcher can eventually move to configuration $\langle a_{27}, b_{24} \rangle$. Right now the contamination status is shown in Figure 5.21 (e)

We keep the first searcher stationary while moving the second one. It is easy to see that after moving through a series of configurations, the second searcher can reach configuration $\langle a_{23}, b_{22} \rangle$, as shown in Figure 5.21 (f). We then keep the second searcher stationary while moving the first one. It is also obvious that after moving through some configurations, the first searcher reaches configuration $\langle a_{26}, b_{19} \rangle$ and the contamination status at this moment is shown in Figure 5.21 (g).

Finally, the first searcher moves to configuration $\langle a_{23}, b_{22} \rangle$ where the second searcher is currently at. The contamination string at this moment contains all "0"s, as shown in Figure 5.21 (h). The 1-hole polygon has been cleared by the two 1-searchers.

On the other hand, the polygon shown in Figure 5.22 is not searchable by two 1-searchers. We show all the configurations for the polygon (we only label the relevant points). We consider the situation where the two searchers move on ∂H. Initially the two searchers are in configuration $\langle a_4, b_2 \rangle$. We set that the first searcher moves in the clockwise direction while the second one moves in the counter-clockwise direction, similar to the above situation.

In the starting search state, the area (its size is zero) between the two beams from the flashlight of the two searchers is clear while the area around them is also contaminated. The general goal of the two searchers is to enlarge this area as large as possible. There are three deadlocks in the polygon. When clearing a deadlock, in order to maintain the existing cleared area, the two searchers need to work together to move through the corresponding configurations. This is shown in Figure 5.21 where the searchers tried to clear the deadlock formed by vertices a_{23} and b_{19}.

We mark three general positions, X, Y and Z in Figure 5.22. Note that these positions

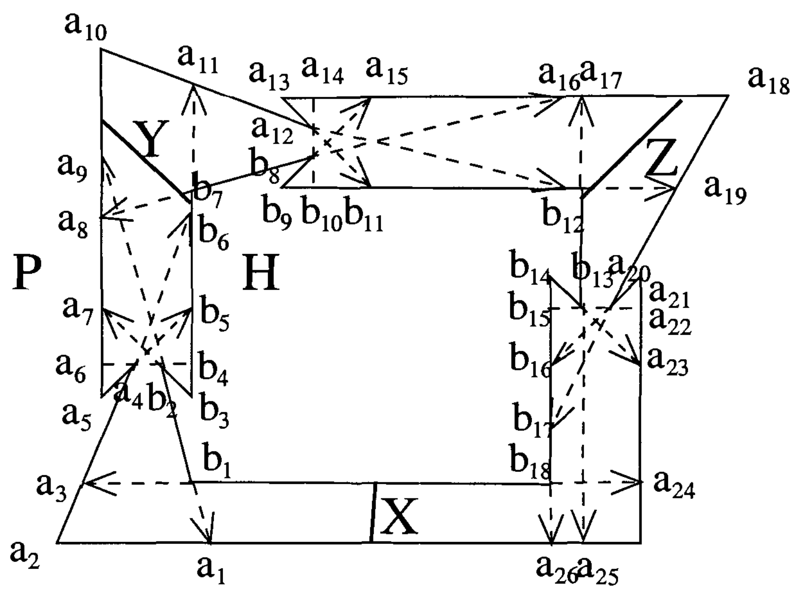


Figure 5.22: The non-searchable 1-hole polygon in Figure 5.5 with relevant configurations indicated.

might not correspond to configurations. After clearing the deadlock formed by vertices a_4 and b_2, the first searcher moves to position Y while the second one moves to position X. The area between the two beams from the flashlights of the searchers in the clockwise direction from the second searcher to the first one is clear while the rest of the polygon is contaminated.

Now in order to clear the deadlock formed by vertices a_{12} and b_8, the first searcher needs to move through the configurations generated by this deadlock. No matter how she tries, the cleared area becomes recontaminated, which invalidates the previous efforts. On the other hand, if the first searcher keeps stationary while the second searcher moves, we have a similar situation where she tries to move through the configurations generated by the deadlock formed by vertices a_{20} and b_{13}. The two searchers might meet in position Z, but all the previous efforts are wasted. They have to start again. This situation means that there is a loop inside graph G_d and there is no path going out of this loop. The two searchers keep moving without making any progress towards the goal that the contamination string contains all "0"s.

It should be pointed out that there can be many starting search states when clearing a polygon by two searchers. But for the particular polygon shown in Figure 5.22, no matter where the searchers start their search, the searchability of the polygon does not change.

Also in general, there could be some loops in search graph G_s. But the existence of such loops is harmless, in terms of the searchability of the polygon, as long as there is path out of them.

5.8 Discussions

We are not satisfied with the result we report here, since we have employed the similar approach proposed by Simov *et al.* [99, 100].

Originally we planned to characterize a 1-hole polygon that is searchable by two 1-searchers. The intuition behind it is based on the observation made regarding Figure 5.6. We have attempted to find a sequence of consecutive deadlocks that satisfy a condition enabling them to be cleared in a collective way. It is easy to see that if there are three such *deadlock sequences* in a 1-hole polygon, then it is not searchable. The deadlock sequences play a similar role to the deadlocks in Figure 5.5. However, it eventually turned out that a precise definition of such a deadlock sequence was really hard to obtain. After having attempted for quite a while, we had to abandon our efforts. But definitely, this will be our next task. We conjecture that the 1-hole polygon search problem by two 1-searchers is easier than its general counterpart without a hole, if we assume weak visibility, and thus it might be easier to obtain the characterization of such a class of searchable polygons.

Before dealing with the 1-hole polygon search problem, we had investigated the room search problem by two 1-searchers, trying to characterize the class of searchable polygons. Unfortunately, our attempts have not succeeded so far.

Chapter 6

Conclusion

Visibility obstruction diagram (VOD) and its variations of a polygon capture the visibility information hidden in a polygon. They represent the inherent relationships between the searchability of a polygon and its geometric characteristics. Previous work using this information, though successful, focused only on the searchability of a polygon. In contrast to this, we have shown that the information provided in the diagram not only helps us decide the searchability of a polygon but also assists us to characterize the class of polygons under different search models. In addition, it is easy to see that, using this information, the characterization of the searchable polygons under different search models is more concise than that in the previous work in terms of the analysis process and understandability.

6.1 Brief summary of polygon search problems

Here we give a brief summary of our current knowledge regarding the polygon search problem. In Table 6.1, each column represents a variant of the polygon search problem while each row corresponds to a search model. The following variants are considered: street (corridor) search problem, room search problem, polygon (no restrictions on the vertices of a polygon) search problem, and 1-hole polygon search problem. We consider the following search models: one ∞-searcher, one boundary 1-searcher model, one non-boundary 1-searcher model (the searcher can move away from the boundary), two guards (with the requirement of mutual visibility) model, two boundary 1-searchers model, and two non-boundary 1-searchers model (the searches can move away from the boundary).

	Street	Room	Polygon	1-hole polygon
One ∞-searcher	Yes (A,C)	Yes (A,C)	Yes (A,C)	Impossible
One boundary 1-searcher	Yes (A,C)	Yes (A,C)	Yes (A,C)	Impossible
One non-boundary 1-searcher	Impossible	Impossible	Impossible	Impossible
Two guards	Yes (A,C)	Yes (A,C)	Yes (A)	Impossible
Two boundary 1-searchers	Yes (A)	Yes (A)	Yes (A)	Yes (A)
Two non-boundary 1-searchers	Yes (A)	Yes (A)	Yes (A)	Impossible

Table 6.1: A brief summary of polygon search problems.

In each cell in the table, *Yes* means the corresponding search problem under the particular model has been solved. *Impossible* shows that the problem cannot be solved under the model. An A inside a cell means that there is a polynomial algorithm for checking the searchability of the polygon under the specific model while a C indicates that a characterization of the class of searchable polygon for the problem has been obtained.

6.1.1 One ∞-searcher model

In many search problems, it has been proved that the power of an ∞-searcher is equivalent to that of a 2-searcher [73, 91, 115].

Crass *et al.* [33] considered the problem of an ∞-searcher searching a corridor. Both an algorithm for checking the searchability and a characterization were obtained.

Lee *et al.* [73] discussed the model for searching a room. Again both a checking algorithm and a characterization were obtained.

As for the simple polygon search problem under this model, Park *et al.* [91] proposed a checking algorithm and a characterization of searchable polygons. Guibas *et al.* [55] obtained a graph-based algorithm for checking the searchability but no characterization was available.

It is impossible for an ∞-searcher to search a 1-hole polygon, since the inner polygon can be considered as an obstacle, and the intruder can always hide from the visibility polygon generated by the flashlight with 360^o visibility.

6.1.2 One boundary 1-searcher model

No previous work has specifically discussed the one boundary 1-searcher for the street search problem. But there are some discussions in [74]. Refer to the discussions in Section 1.3.4.

The papers [74, 108] discussed one boundary 1-searcher in the room search problem. It appears that these two papers developed two different sets of conditions for a room to be searchable. It would be interesting to compare them in terms of the equivalence of the two sets. Both searchability checking algorithms and characterizations were obtained in these work.

The one boundary 1-searcher model was studied originally by Suzuki and Yamashita [106], who first proposed some necessary conditions for the problem.

Park *et al.* [90] studied the problem and so did Tan [109]. However, again, both pieces of work proposed two different sets of conditions. No work has been published for checking if the two sets are equivalent. Both work obtained the searchability checking algorithms and characterizations.

LaValle *et al.* [67, 68] proposed a graph-based algorithm for checking whether a polygon is searchable. However, no characterization was obtained. See the original paper and discussions in Chapters 1 and 2.

As for 1-hole polygons, it is impossible to search them since an intruder can always move just behind or ahead of the flashlight beam. Since the speed of the intruder is alway faster than the speed of the beam rotation and the move of the searcher, there is no way that the intruder can be detected.

6.1.3 One non-boundary 1-searcher model

In this model, in order to distinguish it from the one boundary 1-searcher model, we require that the searcher not always stay on the boundary, i.e. at least for sometime, the searcher have to move into the interior of a polygon.

First we can see that no matter whether the 1-searcher should not always stay on the boundary or not, there is no way for her to clear a polygon with one hole since the intruder can move behind or ahead of the flashlight beam.

For the other three situations, i.e., street search, room search, and general polygon search, even though we put impossible in the three cells in the table, there are still subtle differences, as discussed below.

Recall that we consider two definitions of mutual visibility between two points. In Chapters 2, 3, and 4, we have used the definition which is depicted in Figure 1.2. This definition allows the line segment between two points to graze on a reflex vertex or an edge of the polygon. Under this definition and the requirement from the model, we can still find

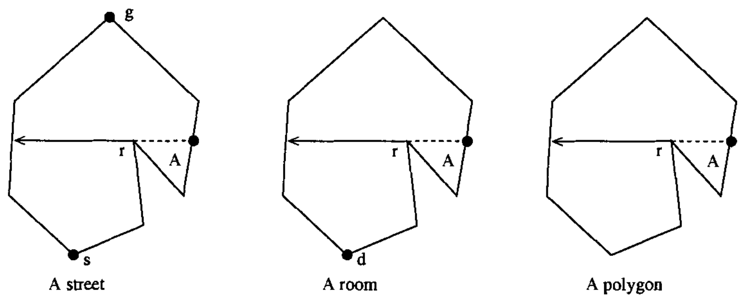


Figure 6.1: Searching under the one non-boundary 1-searcher model.

some streets, rooms and general polygons that are searchable. For instance, we show in Figure 6.1 a searchable street, room and general polygon. Note that at the reflex vertex r, the searcher moves away from the boundary while keeping the beam head stationary. Afterwards, the searcher clears the "pocket" area as indicated by A and the rest of the polygon.

However, in Chapter 5, we have used a different definition, as shown in Figure 5.1. If the line segment between two points graze on a reflex vertex or an edge, then the two points are not mutually visible. It is easy to check that it is impossible for a non-boundary 1-searcher to search the street, the room and the general polygon in Figure 6.1 under this visibility definition and the requirement from the model, since the searcher has to always stay on the boundary.

6.1.4 Two-guard search model

Icking and Klein [57] discussed the two-guard street problem. Different variations were explored. They proposed algorithms for checking whether a street was searchable in each of these variations. They obtained a characterization for each variation. The problem was further discussed by [15, 56, 110] in other directions.

Park *et al.* [89, 93] discussed the two-guard room search problem. They obtained both an algorithm for testing the searchability of a room and a characterization of a searchable room.

As for the general polygon problem, the two guard must clearly stay on the boundary. The only difference of this model from the one boundary 1-searcher model is that no beam head jump is involved. This means that a legal path cannot cross any bone in the SVOD.

Hence the searchability can be easily tested using the SVOD. For instance, by requiring that no bone-crossings be allowed in the SVOD (See Chapters 1, 2 and 4.) of a polygon, we can check whether a polygon is searchable by two guards.

Two guards with the mutual visibility requirement cannot search a polygon with a hole, since any intruder can just follow the beam segment without being detected.

6.1.5 Two boundary 1-searchers model

To our knowledge, no results have been obtained for the two boundary 1-searchers model applied to the street and room search problem. We have applied the model on the room search problem, looking for a characterization of the class of searchable streets and rooms. But it eventually turned out that our attempt was not successful. We consider that by using the approach proposed by Simov *et al.* [99, 100], we could find an algorithm for checking whether a street or a room is searchable under this model. The search graph would be smaller for these two situations than that for the situation where a simple polygon is considered since certain edges in the graph are deleted. However, a characterization of such a class of searchable streets and rooms is still at large.

Simov *et al.* [99, 100] discussed the problem of searching a polygon using two 1-searchers. In their work, the two searchers are not required to stay on the boundary of the polygon. Considering that they converted this setting to the one where both searchers always move on the boundary by introducing *canonical information states*, their result is actually an algorithmic solution to the problem that two boundary 1-searchers search a polygon. But no characterization has been ever obtained.

In this thesis, we have discussed the two boundary 1-searchers for searching a 1-hole polygon in Chapter 5. Originally, we considered that our problem in this setting could be easier than the problem attacked by Simov *et al.* in [99, 100]. But it appeared that this was not the case. We have proposed an algorithmic solution to the 1-hole polygon search problem by two boundary 1-searchers. But again our attempt for a characterization of the class of searchable 1-hole polygons under this setting failed.

6.1.6 Two non-boundary 1-searchers model

We have not seen any work discussing the two 1-searchers model used to search a street or a room. But we consider that the approach proposed by Simov *et al.* is still valid to obtain an algorithm for checking whether a street or a room is searchable by two non-boundary 1-searchers model. We believe that the search graph would be smaller for these two situations than that for the situation where a simple polygon is considered. However, a characterization for the searchable streets or rooms is still not available.

As discussed above, the original problem investigated by Simov *et al.* [99, 100] was for the setting that the two 1-searchers searchers are not required to stay on the boundary of a polygon to search it. They proposed a graph-based algorithmic solution for checking the searchability of a polygon. But no characterization has been obtained.

As for the two non-boundary 1-searchers model on a 1-hole polygon, again we have not seen any result so far.

We believe that as the number of searchers is increased, the complexity of search problem in different settings is increased as well. Sometimes it is extremely hard to conduct a complete and concise analysis for these settings.

6.2 Recent results on visibility obstruction diagram

In our thesis, we have made heavy use of the visibility obstruction diagram to help us tackle different search problems. Since its first proposal [67], there have been some further studies along this direction.

Simov *et al.* [99, 100] attempted to generalize the approach to the search problem by two 1-searchers (not required to be boundary searchers). Lee *et al.* [75] generalized the basic idea of visibility obstruction diagram to the visibility relationships between vertices and edges of a polygon instead of pairs of boundary points. Kameda *et al.* [59] used the diagram to help solve the on-line search problem by a 1-searcher.

It appears that the studies of the diagram for the one boundary 1-searcher model are abundant. However, to the best of our knowledge, generalization of diagram for the two 1-searchers by Simov *et al.* is the farthest attempt at this moment. We believe that as the number of searchers is increased, the complexity of using this type of diagram is greatly increased. A new tool based on a similar idea might be in great need.

6.3 Our contributions

6.3.1 Room search problem

The room search problem is the first problem we investigated in this thesis. Though the problem has been looked at before, the previous analysis is quite lengthy and tedious and is very hard to understand.

As has been shown, the application of VOD to the room search problem reduces this difficulty. We believe that our proof of the characterization of a searchable room by two mutually visible guards is very simple and easy to follow, and it can be readily adapted to deal with the room search problem by a 1-searcher.

We have related the *LR*-visibility of a room and its searchability, making it possible to check the searchability of a room in linear time.

We have further extended the VOD of a room to solve the Find-all-doors problem.

6.3.2 Polygon search problem

We have again successfully applied the VOD of a polygon to investigate the search problem by a 1-searcher. We believe that by making use of the information hidden in the VOD, our characterization of a searchable polygon by a 1-searcher is much easier to understand. The intuition behind the searchability is much more apparent.

In addition, the VOD helped us explore the relationship between a searchable polygon and a searchable room. We have presented an example showing that a searchable polygon does not necessarily contain a door such that the resultant room is searchable.

Unfortunately, our attempt to reduce the time complexity of checking the searchability was not successful. However, we have related the *LR*-visibility and 1-searchability of a polygon. We hope that by relating those two important characteristics of a polygon together, we can take advantage of the nice properties of an *LR*-visible polygon in our further explorations of the polygon search problems.

6.3.3 Searching a 1-hole polygon

Due to the increasing complexity when we increase the number of searchers in our search model, we have started our investigation into a relatively simple but yet still challenging search problem by two 1-searchers. Though we are not successful in characterizing the class of searchable 1-hole polygons, we conjecture that deadlock sequences can shed light on the problem.

6.4 Possible future work

6.4.1 K-link visibility space

Essentially if two boundary points are visible to a common point in a polygon (including one of the two points themselves), we say that those two points are *2*-link visible. This definition can be generalized to the K-link visibility.

Consider a search model in which we have 3 searchers searching a polygon. Two of them are moving along the boundary of the polygon while the third searcher is moving inside the polygon. It is required that the adjacent searchers be mutually visible. It is not difficult to see that if we could represent the 2-link visibility space of the polygon, then the information in this space could help us solve the polygon search problem by 3 searchers.

The situation could be further extended to the search problem by k searchers [41], in which there are k searchers working together to search a polygon.

6.4.2 On-line polygon search problems

If we are searching a polygon for which we have no *à priori* knowledge about its properties, such as the coordinates of its vertices, and lengths of its edges, we search the polygon *on-line*.

On-line polygon search problems have been tried in different directions. However, how to use the VOD of a polygon to help us set down our exploration strategy is probably a new direction [59]. For example, in the proof of Lemma 3.3.3, we construct a legal path. Can we generate a search schedule if we do not know the polygon beforehand? Can we figure out the shape of the room using the information we have obtained during our exploration?

In fact, the same questions are applicable to the polygon search problem by a 1-searcher and the search problem by two 1-searchers.

6.4.3 Randomized polygon search problems

Some previous work has focused on the randomization of the *Art Gallery Problem* [53], though few papers have appeared on this topic in the polygon search problems.

We conjecture that the difficulty of introducing randomization into polygon search problem lies in the fact that both the searcher(s) and intruder(s) are dynamic inside a polygon. There are too many parameters in depicting their pursuit and evasion strategies. However, there is already some work published [13, 53] and we are starting working in this direction.

Bibliography

[1] J. Abello, V. Estivill-Castro, T. Shermer, and J. Urrutia. Illuminating with orthogonal floodlights. In *Algorithms and Computations*, number 1004 in Lecture Notes in Computer Science, pages 362–371. Springer-Verlag, 1995.

[2] S. Albers and M. R. Henzinger. Exploring unknown environments. In *Proceedings of the ACM Symposium on Theory of Computing STOC 1997*, pages 416–425, 1997.

[3] D. Angluin, J. Westbrook, and W. Zhu. Robot navigation with range queries. In *Proceedings of the ACM Symposium on Theory of Computing (STOC 1996)*, pages 469–478, 1996.

[4] B. Aronov, A. R. Davis, T. K. Dey, S. P. Pal, and D. C. Prasad. Visibility with reflection. In *Proceedings of the 11th ACM Symposium on Computational Geometry 1995*, pages 316–325, 1995.

[5] B. Aronov, A. R. Davis, T. K. Dey, S. P. Pal, and D. C. Prasad. Visibility with one reflection. *Discrete and Computational Geometry*, 19:553–574, 1998.

[6] R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins. Searching in the plane. *Information and Computation*, 106(2):234–252, 1993.

[7] J. Bagga, L. Gewali, S. Ntafos, and J. Urrutia. Internal, external, and mixed visibility edges of polygons (preprint), 1997.

[8] E. Bar-Eli, P. Berman, A. Fiat, and P. Yan. Online navigation in a room. *Journal of Algorithms*, 17(3):319–341, 1994.

[9] P. Belleville, P. Bose, J. Czyowicz, J. Urrutia, and J. Zaks. K-guarding polygons on the plane. In *Proceedings of the 6th Canadian Conference on Computational Geometry 1994*, pages 381–386, 1994.

[10] M. A. Bender, A. Fernndez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC) 1998*, pages 269–278, 1998.

[11] J. L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transaction on Computers*, C-28:643–647, 1979.

[12] M.de Berg, M.van Kreveld, M.Overmars, and O.Schwarzkopf. *Computational Geometry (Algorithms and Applications)*. Springer-Verlag, August 1998.

[13] P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosn, and M. Saks. Randomized robot navigation algorithms. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms 1996*, pages 75–84, 1996.

[14] B. K. Bhattacharya and S. K. Ghosh. Characterizing LR-visibility polygons and related problems. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, 1998.

[15] B.K. Bhattacharya, A. Mukhopadhyay, and G. Narasimhan. Optimal algorithms for two-guard walkability of simple polygons. In F. K. H. A. Dehne, J. Sack, and R. Tamassia, editors, *Algorithms and Data Structures, 7th International Workshop, WADS 2001, Providence, RI, USA, August 8-10, 2001, Proceedings*, volume 2125 of *Lecture Notes in Computer Science*, pages 438–449. Springer, 2001.

[16] T. Biedl, E. Demaine, M. Demaine, A. Lubiw, and G. Toussaint. Hiding disks in folded polygons. In *Proceedings of the Tenth Canadian Conference on Computational Geometry 1998*, 1998.

[17] D. Bienstock and P. Seymour. Monotonicity in graph searching. *Journal of Algorithms*, 12(2):239–245, June 1991.

[18] P. Bose. Visibility in simple polygons. Master's thesis, University of Waterloo, Ontario, Canada, Ontario, Canada, 1991.

[19] P. Bose, A. Dean, J. Hutchinson, and T. Shermer. On rectangle visibility graphs. In *Proceedings of Graph Drawing*, pages 25–35, 1997.

[20] P. Bose, L. Guibas, A. Lubiw, M. Overmars, D. Souvaine, and J. Urrutia. The floodlight problem. *International Journal of Computational Geometry and Applications*, 7:153–163, 1997.

[21] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. In *Proceedings of the Fourth Canadian Conference on Computational Geometry 1992*, pages 23–28, 1992.

[22] B. Broden, M. Hammar, and B. J. Nilsson. Guarding lines and 2-link polygons is apx-hard. In *Proceedings of Thirteenth Canadian Conference on Computational Geometry 2001*, pages 45–48, 2001.

[23] S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. In *Proceedings of International Symposium of Algorithms and Computation 1993*, number 762 in Lecture Notes in Computer Science, pages 58–67. Springer-Verlag, 1993.

[24] S. Carlsson, H. Jonsson, and B.J. Nilsson. Approximating the shortest watchman route in a simple polygon. Technical Report Technical Report LU-CS-TR:97-190, Department of Computer Science, Lund University, 1997.

[25] S. Carlsson and B. J. Nilsson. Computing vision points in polygons. *Algorithmica*, 24(1):50–75, 1999.

[26] S. Carlsson, B. J. Nilsson, and S. C. Ntafos. Optimum guard covers and m-watchmen routes for restricted polygons. *International Journal of Computational Geometry and Applications*, 3(1):85–105, 1993.

[27] B. Chazelle and L.J. Guibas. Visibility and intersection problems in plane geometry. In *Proceedings of the Third Annual Symposium on Computational Geometry*, pages 135–146, 1985.

[28] O. Cheong and R. van Oostrum. The visibility region of points in a simple polygon. In *Electric Proceedings of 11th Canadian Conference on Computational Geometry 1999*, 1999.

[29] W. P. Chin and S. Ntafos. Optimum watchman routes. In *Proceedings of the 2nd ACM Annual Symposium on Computational Geometry 1986*, pages 24–33, 1986.

[30] W.P. Chin. The zookeeper route problem. *Information Sciences*, 63:245–259, 1992.

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

[32] P. Colley, H. Meijer, and D. Rappaport. Motivating lazy guards. In *Proceedings of Canadian Conference in Computational Geometry 1995*, pages 121–126, 1995.

[33] D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor: the open edge variant of the polygon search problem. *International Journal of Computational Geometry and Applications*, 5(4):397–412, 1995.

[34] J. Czyzowicz, E. Rivera-Campo, and J. Urrutia. Optimal floodlight illumination of stages. In *Proceedings of the Fifth Canadian Conference on Computational Geometry 1993*, pages 393–398, 1993.

[35] V. Estivill-Castro D. Chen and J. Urrutia. Optimal guarding of polygons and monotone chains. In *Proceedings of Seventh Canadian Conference on Computational Geometry 1995*, pages 133–138, 1995.

[36] G. Das, P. J. Heffernan, and G. Narasimhan. LR-visibility in polygons. *Computational Geometry*, 7:37–57, 1997.

[37] A. Datta and C. Icking. Competitive searching in a generalized street. *Computational Geometry*, 13(2):109–120, 1999.

[38] M. de Mark, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications.* Springer-Verlag, second edition, 2000.

[39] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment (extended abstract). In *Proceedings of the Symposium on Foundations of Computer Science 1991*, pages 298–303, 1991.

[40] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *Journal of the ACM*, 45(2):215–245, March 1998.

[41] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali. Sweeping simple polygons with a chain of guards. In *Proceedings of the 11th Annual ACMSIAM Symposium on Discrete Algorithms 2000*, pages 927–936, 2000.

[42] S. Eidenbenz. Inapproximability results for guarding polygons without holes. In *Proceedings of International Symposium of Algorithms and Computation 1998*, number 1533 in Lecture Notes in Computer Science, pages 427–436. Springer-Verlag, 1998.

[43] S. Eidenbenz. How many people can hide in a terrain? In *Proceedings of International Symposium of Algorithms and Computation 1999*, number 1741 in Lecture Notes in Computer Science, pages 184–194. Springer-Verlag, 1999.

[44] S. Eidenbenz and C. Stamm. Maximum clique and minimum clique partition in visibility graphs. In *Proceedings of IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, number 1872 in Lecture Notes in Computer Science, pages 200–212. Springer-Verlag, 2000.

[45] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability of some art gallery problems. In *Proceedings of the Tenth Canadian Conference on Computational Geometry 1998*, pages 427–436, 1998.

[46] V. Estivill-Castro, J. O'Rourke, J. Urrutia, and D. Xu. Illumination of polygons with vertex floodlights. *Information Processing Letters*, 1(56):62–73, 1995.

[47] V. Estivill-Castro and J. Urrutia. Optimal floodlight illumination of orthogonal art galleries. In *Proceedings of the Sixth Canadian Conference in Computational Geometry 1994*, pages 81–86, 1994.

[48] V. Estivill-Castro and J. Urrutia. Two-floodlight illumination of convex polygons. In *Proceedings of the Fourth Workshop on Algorithms and Data Structures WADS-95 Kingston*, number 955 in Lecture Notes in Computer Science, pages 62–73, Ontario, Canada, 1995. Springer-Verlag.

[49] S. Fisk. A short proof of chvátal's watchman theorem. *Journal of Combinatorial Theory B*, 24:374, 1978.

[50] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[51] L. Gewali and S. Ntafos. Watchman routes in the presence of a pair of convex polygons. *Journal of Information Sciences*, 105:123–149, 1998.

[52] S. K. Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete and Computational Geometry*, 17(2):143–162, March 1997.

[53] H. Gonzlez-Banos and J. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the ACM Symposium on Computational Geometry 2001*, pages 232–240, 2001.

[54] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[55] L.J. Guibas, J.C. Latombe, S.M. LaValle, D. Lin, and R. Motwani. A visibility-based pursuit-evasion problem. *International Journal of Computational Geometry and Applications*, 9(5):471–494, October 1999.

[56] P. Heffernan. An optimal algorithm for the two-guard problem. *International Journal of Computational Geometry and Applications*, 6:15–44, 1996.

[57] C. Icking and R. Klein. The two guards problem. *International Journal of Computational Geometry and Applications*, 2(3):257–285, 1992.

[58] C. Icking, R. Klein, and E. Langetepe. An optimal competitive strategy for walking in streets. In *Proceedings of the Sixteenth Symposium of Theoretical Aspects of Computer Science 1999*, number 1563 in Lecture Notes in Computer Science, pages 110–120. Springer-Verlag, 1999.

[59] T. Kameda, M. Yamashita, and I. Suzuki. On-line polygon search by a six-state boundary 1-searcher. Technical Report CS/TR-2003-07, Simon Fraser University, October 2003.

[60] J. M. Kleinberg. On-line search in a simple polygon. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 8–15, 1994.

[61] E. Kranakis, D. Krizanc, and J. Urrutia. On the number of directions in visibility representations of graphs. In *Proceedings of International Workshop on Graph Drawings*, number 894 in Lecture Notes in Computer Science, pages 167–176. Springer-Verlag, 1994.

[62] A. Kndgen. Art galleries with interior walls. *Discrete and Computational Geometry*, 22:249–258, 1999.

[63] A. S. LaPaugh. Recontamination does not help to search a graph. *Journal of the ACM*, 40(2):224–245, April 1993.

[64] S. M. LaValle, H. H. Gonzlez-Baos, C. Becker, and J. C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proceedings of IEEE International Conference on Robotics and Automation 1997*, pages 731–736, 1997.

[65] S. M. LaValle and J. Hinrichnsen. Visibility-based pursuit-evasion: The case of curved environments. In *Proceedings of IEEE International Conference on Robotics and Automation 2002*, pages 1671–1677, 1999.

[66] S. M. LaValle, D. Lin, L. J. Guibas, J. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proceedings of IEEE International Conference on Robotics and Automation 1997*, pages 737–742, 1997.

[67] S. M. LaValle, B. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. In *Proceedings of the ACM Symposium on Computational Geometry 2000*, pages 260–269, Hong Kong University of Science and Technology, Hong Kong, 2000.

[68] S. M. LaValle, B. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. *International Journal of Computational Geometry and Applications*, 12(1-2):87–113, 2002.

[69] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32:276–282, March 1986.

[70] 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.

[71] D.T. Lee. *The Computer Science and Engineering Handbook*, chapter Computational Geometry, pages 111–140. CRC Press, 1996.

[72] J. H. Lee, S. M. Park, and K. Y. Chwa. On the polygon-search conjecture. Technical Report Technical Report KAIST CS/TR-2000-156, Korea Advanced Institute of Science and Technology(KAIST), October 2000.

[73] J. H. Lee, S. Y. Shin, and K. Y. Chwa. Visibility-based pursuit-evasions in a polygonal room with a door. In *Proceedings of the ACM Symposium on Computational Geometry 1999*, pages 281–290, 1999.

[74] J.H. Lee, S. M. Park, and K. Y. Chwa. Searching a polygonal room with one door by a 1-searcher. *International Journal of Computational Geometry and Applications*, 10(2):201–220, 2000.

[75] J.H. Lee, S. M. Park, and K. Y. Chwa. Simple algorithms for searching a polygon with flashlights. *Information Processing Letters*, 81:265–270, 2002.

[76] T. Lengauer and R. E. Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *Journal of the ACM*, 29(4):1087–1130, October 1982.

[77] N. Megiddo, S. L. Hakimi, M.R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18–44, 1988.

[78] S. Ntafos. The robber route problem. *Information Processing Letters*, 34(2):59–63, 1990.

[79] S. Ntafos. Watchman routes under limited visibility. In *Proceedings of the 2nd Canadian Conference on Computatial Geometry 1990*, pages 89–92, 1990.

[80] S. Ntafos and L. Gewali. External watchman routes. *The visual computer*, 10:474–483, 1994.

[81] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, August 1987.

[82] J. O'Rourke. Open problems in the combinatorics of visibility and illumination. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 237–243. (Contemporary Mathematics) American Mathematical Society, 1998.

[83] J. O'Rourke, T. Shermer, and I. Streinu. Illuminating convex polygons with vertex floodlights. In *Proceedings of the 7th Canadian Conference on Computational Geometry 1995*, pages 151–156, August 1995.

[84] J. O'Rourke and I. Streinu. Visibility in pseudo-polygons and vertex-edge pseudo-visibility graphs. In *Electronic Proceedings of the Fifth MSI-Stony Brook Workshop on Computational Geometry, 1995*, 1995.

[85] J. O'Rourke and I. Streinu. Vertex-edge pseudo-visibility graphs: Characterization and recognition. In *Proceedings of the ACM Symposium on Computational Geometry 1997*, pages 119–128, 1997.

[86] J. O'Rourke and I. Streinu. The vertex-edge visibility graph of a polygon. *Computational Geometry: Theory and Applications*, 10:105–120, 1998.

[87] P. Panaite and A. Pelc. Exploring unknown undirected graphs (preliminary version). In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms 1998*, pages 316–322, 1998.

[88] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33:281–295, 1999.

[89] S. M. Park, J. H. Lee, and K. Y. Chwa. Characterization of rooms searchable by two guards. In *Proceedings of International Symposium on Algorithms and Computation 2000*, pages 515–526, 2000.

[90] S. M. Park, J. H. Lee, and K. Y. Chwa. A characterization of the class of polygons searchable by a 1-searcher. Technical Report CS/TR-2000-160, Korea Advanced Institute of Science and Technology, December 2000.

[91] S. M. Park, J. H. Lee, and K. Y. Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. Technical Report Technical Report KAIST CS/TR-2000-161, Korea Advanced Institute of Science and Technology(KAIST), January 2001.

[92] S. M. Park, J. H. Lee, and K. Y. Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming 2001*, pages 456–468, 2001.

[93] S. M. Park, Jae-Ha Lee, and Kyung-Yong Chwa. Searching a room by two guards. *International Journal of Computational Geometry and Applications*, 12(4):339–352, 2002.

[94] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. Lick, editors, *Theory and Applications of Graphs*, Lecture Notes in Mathematics, pages 426–441. Springer-Verlag, 1976.

[95] S. Rajko and S. M. LaValle. A pursuit-evasion bug algorithm. In *Proceedings of IEEE International Conference on Robotics and Automation 2001*, pages 1954–1960, 2001.

[96] S. Sachs, S. Rajko, and S. M. LaValle. Visibility-based pursuit-evasion in an unknown planar environment. In *Submitted to International Journal of Robotics Research*, 2002.

[97] S. Schuierer and D. Wood. Multiple-guard kernels of simple polygons. *Journal of Geometry*, 66:161–186, 1999.

[98] T. Shermer. Recent results in art galleries (invited paper). In *Proceedings of the IEEE, 80(9)*, pages 1384–1399, 1992.

[99] B. Simov, S. M. LaValle, and G. Slutzki. A complete pursuit-evasion algorithm for two pursuers using beam detection. In *Proceedings of IEEE International Conference on Robotics and Automation 2002*, pages 618–623, 2002.

[100] B. Simov, G. Slutzki, and S. M. LaValle. Clearing a polygon with two 1-searchers. Technical report, Iowa State University.

[101] B. Simov, G. Slutzki, and S. M. LaValle. Pursuit-evasion using beam detection. In *Proceedings of IEEE International Conference on Robotics and Automation 2000*, pages 1657–1662, 2000.

[102] W. Steiger and I. Streinu. Illumination by floodlights. *Computational Geometry: Theory and Applications*, 10:57–70, 1998.

[103] I. Streinu. Non-stretchable pseudo-visibility graphs. In *Electric Proceedings of 11th Canadian Conference on Computational Geometry 1999*, 1999.

[104] K. Sugihara, I. Suzuki, and M. Yamashita. The searchlight scheduling problem. *SIAM Journal on Computing*, 19(6):1024–1040, 1990.

[105] I. Suzuki, Y. Tazoe, M. Yamashita, and T. Kameda. Searching a polygonal region from the boundary. *International Journal of Computational Geometry and Applications*, 11(5):529–553, 2001.

[106] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, October 1992.

[107] I. Suzuki, M. Yamashita, H. Umemoto, and T. Kameda. Bushiness and a tight worst-case upper bound on the search number of a simple polygon. *Information Processing Letters*, 66(1):49–52, 1998.

[108] X. Tan. Efficient algorithms for searching a polygonal room with a door. In *Discrete and Computational Geometry, Japanese Conference, JCDCG 2000*, pages 339–350, 2000.

[109] X. Tan. Searching a simple polygon by a k-searcher. In *Proceedings of International Symposium on Algorithms and Computation 2000*, pages 503–514, 2000.

[110] L. H. Tseng, P. J. Heffernan, and D. T. Lee. Two-guard walkability of simple polygons. *International Journal of Computational Geometry and Applications*, 8(1):85–116, 1998.

[111] J. Urrutia. On the number of internal and external visibility edges of polygons. Technical Report Technical report TR-97-04, Department of Computer Science, University of Ottawa, February 1997.

[112] J. Urrutia. Sixth proof of the orthogonal art gallery theorem. Technical Report Technical report TR-97-03, Department of Computer Science, University of Ottawa, February 1997.

[113] J. Urrutia. *Handbook of Computational Geometry (Editors: J. R. Sack and J. Urrutia)*, chapter Art Gallery and Illumination Problems, pages 973–1027. North-Holland, Amsterdam, Netherlands, 2000.

[114] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers (extended abstract). In *Proceedings of the ACM Symposium on Computational Geometry 1997*, pages 448–450, 1997.

[115] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers. *Algorithmica*, 31(2):208–236, 2001.