1-1-1995

# Visibility properties of polygons

David Bruce Glasser
*University of Nevada, Las Vegas*

# INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

# VISIBILITY PROPERTIES OF POLYGONS

by

David Bruce Glasser

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Science

Department of Computer Science
University of Nevada, Las Vegas
December, 1995

The Thesis of David Bruce Glasser for the degree of Master in Computer Science is approved.

L. P Gewali 11-15-95

Chairperson, Laxmi Gewali, PhD.

11-15-95

Examining Committee Member, Evangelos Yfantis, PhD.

Ajoy Kumar Datta

Examining Committee Member, Ajoy Datta, PhD.

11-20-95

Graduate Faculty Representative, Peter Shiue , PhD.

C L Bowler 12-4-95

Interim Dean of the Graduate College, Cheryl L. Bowler, EdD.

University of Nevada, Las Vegas
December, 1995

## ABSTRACT

Two problems dealing with visibility in the interior of a polygon are investigated. We present a linear time algorithm for computing the stair-case visibility polygon from a point inside a simple polygon, which is optimal within a constant factor. We show that the problem of locating the minimum number of $90^0$-flood-lights to illuminate the interior of a simple polygon is NP-complete. We also discuss the generalization of the above results.

TABLE OF CONTENT

List of Figures

## Acknowledgements

I would like to thank Dr. Laxmi Gewali for his continual help and input during the course of studying these problems and preparing this thesis. In addition I would like to thank Dr. Evangelos Yfantis, Dr. Ajoy Datta, and Dr. Peter Shiue for reviewing this thesis and participating in its review and for their constant encouragement. I would also like to thank my parents for their many years of support and encouragement to excel in whatever I choose to pursue. This thesis is dedicated to my wife and proofreader, Neeshma, for her help and support in completing this thesis.

# Chapter One

## Introduction

The notion of visibility has been useful for solving problems in computational geometry. Under the standard notion of visibility, two points inside a polygon are said to be **visible** if the line segment connecting them does not intersect with the exterior of the polygon. Many interesting problems dealing with visibility properties of polygons have been presented as art gallery problems [1,2]. The standard art gallery problem asks for locating a minimum set of point guards to be placed in the interior of a polygon such that each point inside the polygon is visible to at least one guard. The standard art gallery problem is known to be NP-hard [1]. Approximation algorithms for solving the standard art gallery problem have been developed. For example, it has been established that a simple polygon of $n$ sides can always be guarded by $\lfloor n/3 \rfloor$ point guards [1].

An interesting problem, which can be viewed as a simplified version of the standard art gallery problem, is the computation of the area visible to a guard from a given point inside a polygon. The area visible from a point inside a polygon is called the **visibility polygon.** Linear time algorithms for computing the visibility polygon from a point have been reported [3,4]. Since the size of a visibility polygon can be of the same order of the size of the polygon itself, any linear time algorithm for computing a visibility polygon is optimal within a constant factor. Figure 1.1 shows an example of a visibility polygon.

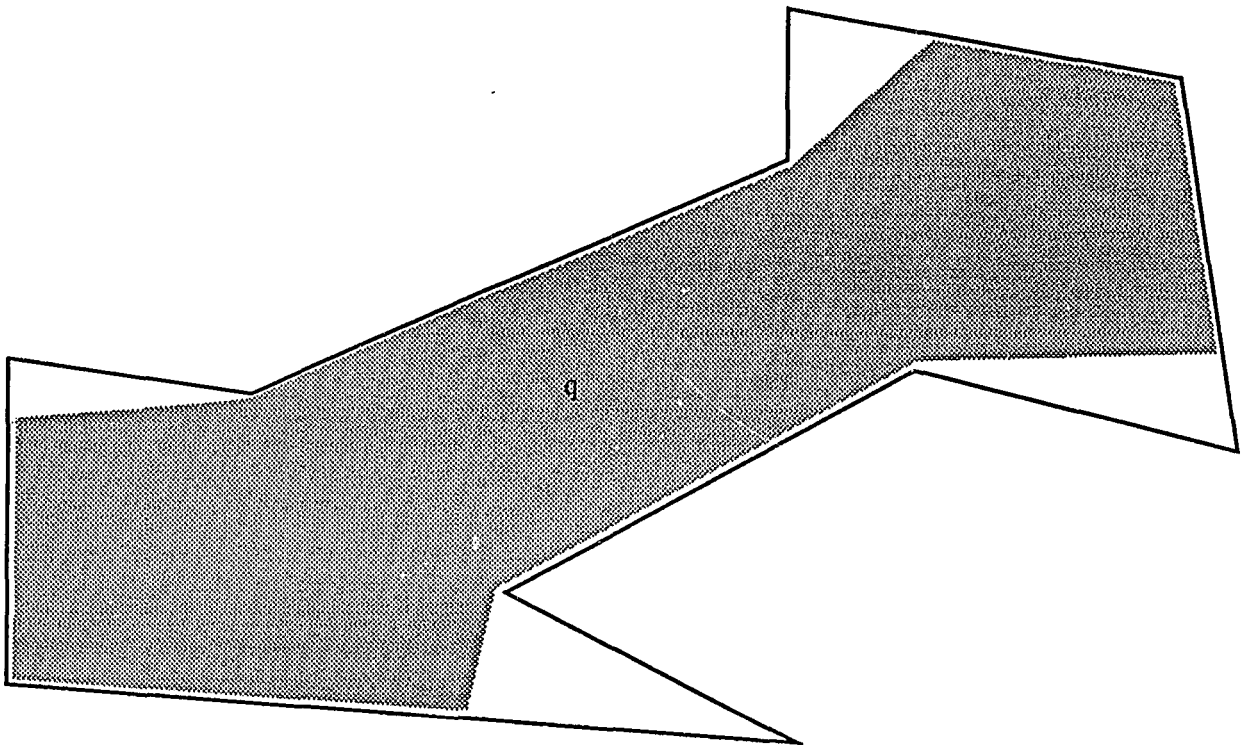



Figure 1.1: Illustrating the visibility polygon from a point q

---

Another interesting problem, which can be viewed as a restricted form of the standard art gallery problem, is the single guard placement problem that asks for the placement of a point guard inside a polygon to maximize visibility. Figure 1.2 shows an example placement that maximizes the area of a visibility polygon. Approximation algorithms for solving the single guard placement problem have been reported [5] and the existence of an exact algorithm is open.

Many interesting variations on the standard notion of visibility have been pursued. One such variation is the notion of **s-visibility**. Two points inside a polygon are said to be **s-visible** if there exists a stair-case path connecting them that does not intersect with the exterior of the polygon. S-visibility properties have been considered mostly for orthogonal polygons [6,7,8]. The notion of s-visibility has been useful in giving insight to the decomposition of polygons. While the problem of decomposing a simple orthogonal polygon into a minimum number of star polygons is known to be NP-hard, the same problem can be solved in polynomial time under s-visibility [6]. The algorithm for decomposing an orthogonal polygon into s-visibility polygons has time complexity $O(n^8)$, where $n$ is the number of vertices of the polygon; this time complexity is rather high for practical use and the development of a faster algorithm is still open.

Another notion of visibility closely related to s-visibility is **r-visibility**. Two points inside an orthogonal polygon are said to be **r-visible** if they are the end points of the diagonal of an isothetic rectangle which lies completely inside the polygon. The notion of r-visibility has been used to decompose a horizontally convex polygon into simple shapes [9].

The computation of the visibility area from points along a line segment has been considered; these problems are referred to as **weak visibility** problems in the literature [1,10,11]. A point $p$ inside a polygon is said to be **weakly visible** from a line segment $l$ if $p$ is visible from some points in $l$. On the other hand, if point $p$ is visible from every point on $l$, then $p$ is said to be **strongly visible** from $l$. Efficient algorithms for recognizing weakly visible polygons have also been developed [10].

The notion of visibility is closely related to the notion of shortest collision-free paths. It turns out that the shortest path connecting two given points consists of visibility edges between vertices. The set of all visibility edges of a polygon is called

its **visibility graph**. The visibility graph is precisely the union of the set of shortest paths connecting all pairs of vertices [1,2]. One of the most outstanding open problems of visibility is the recognition of a visibility graph: given a graph $G$, determine if $G$ is a visibility graph or not [1].

In this thesis, we address two problems on visibility. In chapter two, we develop an algorithm to compute a visibility polygon from a point inside a polygon under stair-case visibility. The algorithm runs in $O(n)$ time, which is optimal within a constant factor. In chapter three, we address the problem of illuminating polygons by $90^0$-flood-lights. Under $\alpha$-flood-light aperture, visibility is restricted to $\alpha$-degrees from any single source. We show that computing the minimum number of $90^0$-flood-lights to illuminate a simple polygon is NP-complete. The reduction is based on the construction of a polygon that represents a satisfiability problem. In chapter four, we discuss further research and interesting unresolved problems.

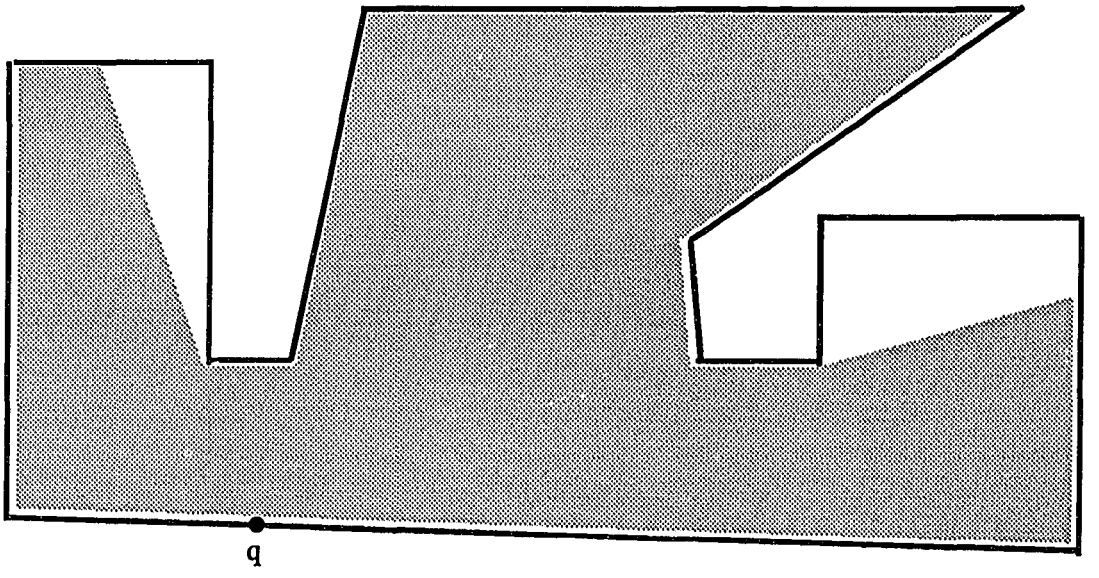



Figure 1.2: Illustrating the maximum visibility area for a single point guard

# Chapter Two

## Computing Visibility Polygons Under Stair-Case Visibility

### 2.1 Introduction

In this chapter, we consider the problem of computing visibility polygons under stair-case visibility. When dealing with stair-case visibility, it is natural to consider the underlying polygon to be orthogonal. Note that a polygon is orthogonal (or isothetic) if its edges are parallel to the coordinate axes. A simple path consisting of horizontal and vertical line segments is called an orthogonal path. A stair-case path is an orthogonal path whose projections along the x-axis and the y-axis do not overlap (see Figure 2.1). In other words, a stair-case path is monotone along both the x-axis and y-axis. We define two points inside an orthogonal polygon to be **s-visible** if they can be connected by a stair-case path without intersecting the exterior of the polygon. The notion of s-visibility naturally leads to the notion of an s-visibility polygon. The s-visibility polygon from a point $q$ inside a polygon $Q$ is the set of points that are s-visible from $q$. Figure 2.2 shows an example of an s-visibility polygon.

The notion of s-visibility has been useful in gaining insight to the problem of decomposing orthogonal polygons into simple shapes. It has been established that the visibility graph of a simple orthogonal polygon under s-visibility is weakly triangulated [6,7,8]. This structural property of the s-visibility graphs has been used to develop polygonal time algorithms to decompose orthogonal polygons into s-star polygons.

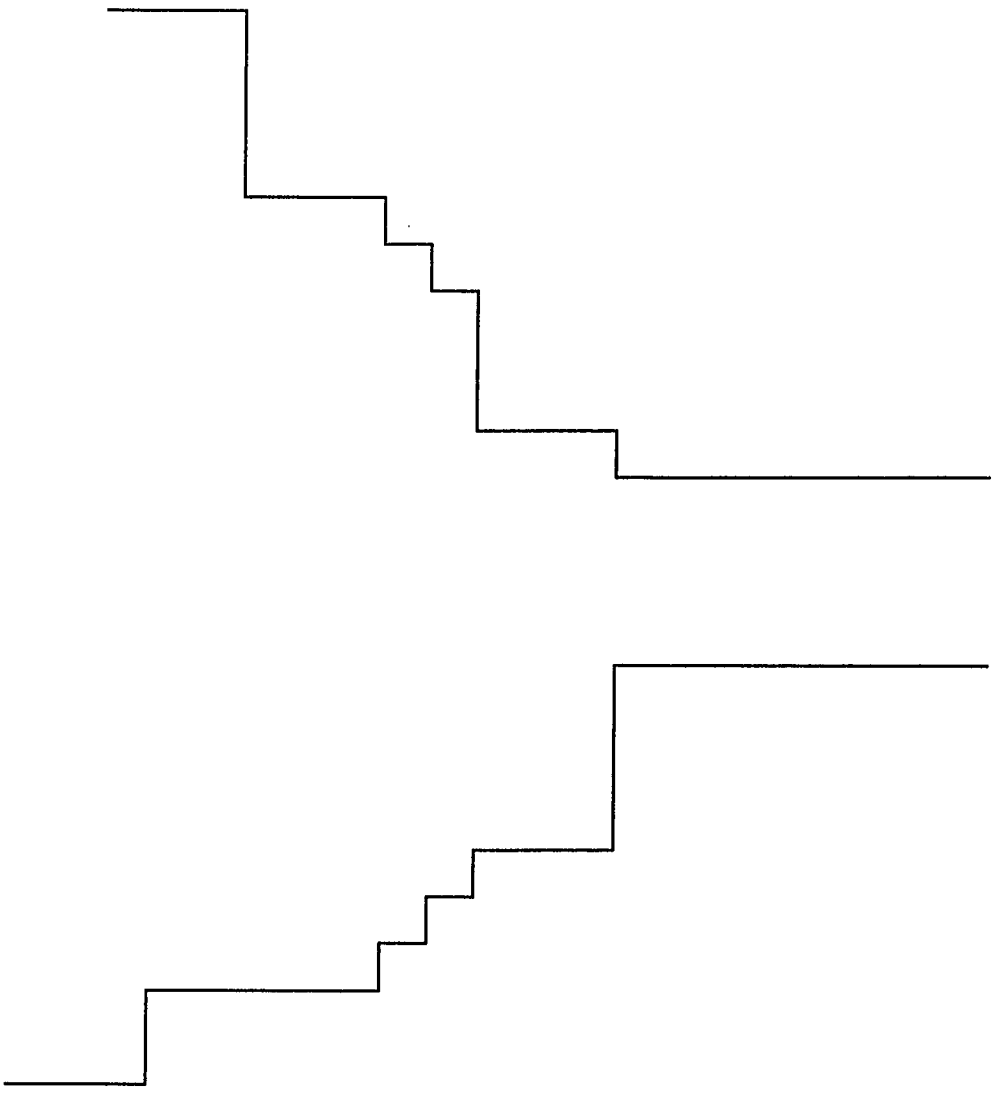


Figure 2.1: Illustrating stair-case paths

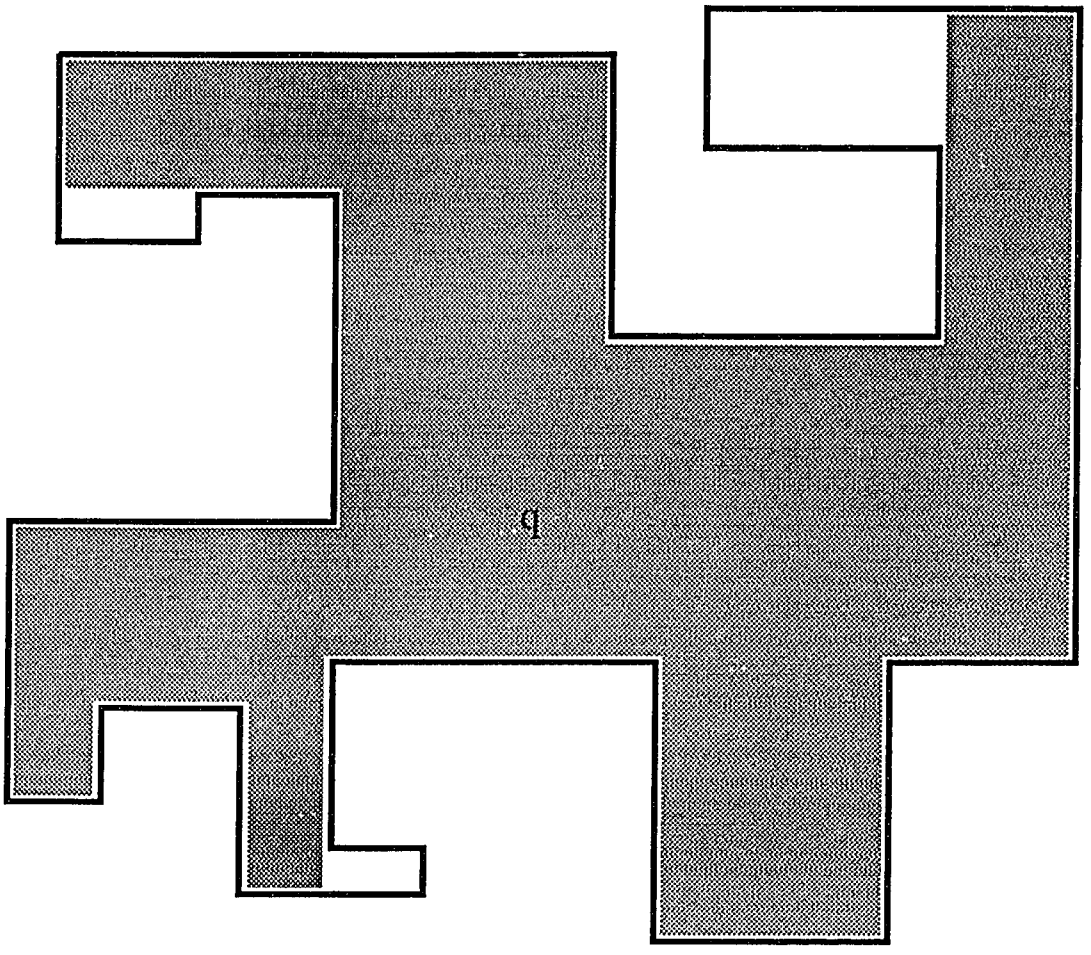



Figure 2.2: Illustrating a stair-case visibility polygon

## 2.2 Preliminaries

The input polygon $Q$ is given as an ordered sequence of vertices, $v_0, v_1, v_2, v_3, ..., v_{n-1}$, in the order in which they occur along the clockwise traversal of its boundary. A path is said to be monotone along a given direction $d$ if any line segment perpendicular to $d$ intersects the path in at most one point. An orthogonal path having monotonicity

along the x-axis direction is called an **x-monotone path.** Similarly, a **y-monotone path** can be defined. Visibility polygons from a point can be defined in terms of x-monotone visibility and y-monotone visibility. An **x-monotone visibility polygon,** from a point $q$, is the set of points that can be connected to $q$ by an x-monotone path. A **y-monotone visibility polygon** can be defined similarly. Let $V_q^s(Q)$, $V_q^x(Q)$, and $V_q^y(Q)$ denote the s-visibility polygon, x-monotone visibility polygon, and y- monotone visibility polygon, respectively, from a point $q$ inside polygon $Q$.

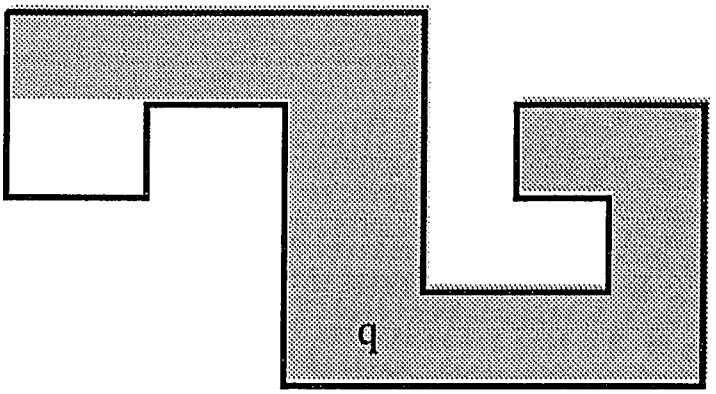
**Lemma 2.1:** *The s-visibility polygon $V_q^s(Q)$ is given by the intersection of $V_q^x(Q)$ and $V_q^y(Q)$.* (see Figure 2.3.)

## 2.3 The Algorithm

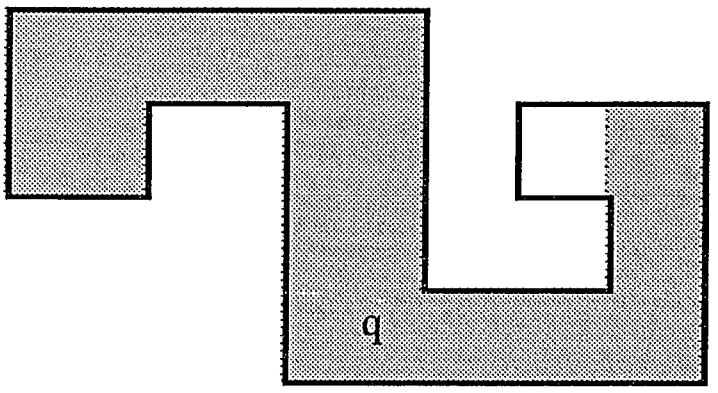
In this section we demonstrate an algorithm for computing the s-visibility polygon from a given point inside a simple orthogonal polygon. Establishing that the s-visibility polygon from a point in the interior of a simple orthogonal polygon is a simple orthogonal polygon is straightfoward.

**Lemma 2.2:** *$V_q^s(Q)$, the s-visibility polygon from a point $q$ inside an orthogonal polygon $Q$, is an orthogonal polygon.*

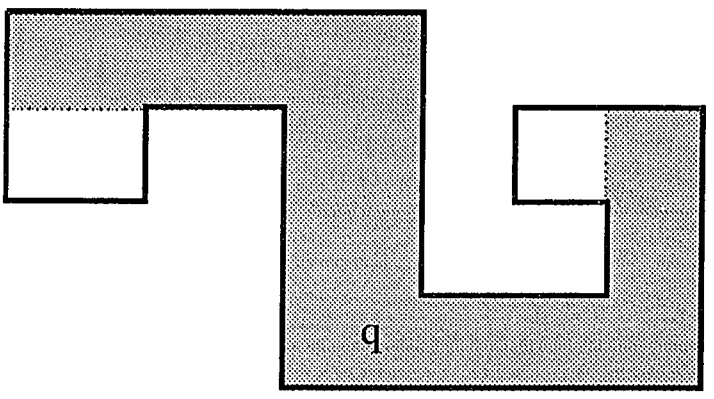
**Proof:** Assume the contrary that an s-visibility polygon is not orthogonal. Then some edge $(a, b)$ on the boundary of the s-visibility polygon $V$ is neither vertical nor horizontal. Consider a visibility path connecting $q$ to an interior point $x$ in the edge $(a, b)$. Since the edge $(a, b)$ lies completely inside the polygon $Q$, the s-visibility path connecting $q$ to $x$ can be extended beyond $x$ to the outside of the visibility polygon $V$. This implies that a point lying outside of the visibility polygon $V$ is s-visible to $q$ - a contradiction. □




(a) x-monotone visibility polygon




(b) y-monotone visibility polygon




(c) resulting s-visibility polygon

Figure 2.3: Illustrating s-visibility polygon components

An overview of the algorithm can be briefly stated as follows: (i) We first compute $R = V_q^x(Q)$, the x-monotone visibility polygon from point $q$ for polygon $Q$. (ii) We then compute the y-monotone visibility polygon from point $q$ for polygon $R$, which is precisely the required s-visibility polygon. To compute the x-monotone visibility polygon efficiently, we partition the polygon into four parts by extending horizontal and vertical lines through $q$ until they meet the boundaries of the polygon (see Figure 2.4). We call these four parts **NE-part, NW-part, SE-part,** and **SW-part,** with obvious meanings. In what follows, we describe the computation of the x-monotone visibility polygon for the NE-part. Computations for the other parts are similar. We partition the NE-part into trapezoids by extending the vertical edges of the polygon to its interior. Since the polygon is orthogonal, all trapezoids are rectangles. We call two rectangles neighbors if they share a vertical chord. A neighbor can either be an east neighbor (**E-neighbor**) or a west neighbor (**W-neighbor**).

We start with the unique rectangle of the NE-part that has $q$ as its vertex as the partially constructed x-monotone visibility polygon. As the algorithm proceeds, other rectangles are added, one at a time, by examining the neighbors of the rectangles lying on the boundary of the partially constructed s-visibility polygon. The partially constructed solution is expanded by adding rectangles lying to the east of its boundary. The process of expansion stops when no more expansions can be made. We can implement this mechanism by using a simple queue data structure. First we enque the unique rectangle with $q$ as one of its vertices. Then we dequeue a rectangle $r$ from the queue and this becomes the currently selected candidate to be added to the partially constructed solution. Neighbors of $r$ lying to the east are added to the queue and the currently examined rectangle is added to the partial solution. This process is continued until the queue is empty and the resulting solution is the x-monotone visibility polygon for the NE-part. Similarly, the x-monotone visibility polygons for

the remaining three parts are computed and $R = V_q^x(Q)$ is determined. Next, we compute the y-monotone visibility polygon from point $q$ for polygon $R$, which precisely gives the s-visibility polygon. A formal description of the S-Visibility Polygon Algorithm is given below.

**S-Visibility Polygon Algorithm**

**Input:** A simple orthogonal polygon $Q$ with a given point $q$ inside it.

**Output:** The s-visibility polygon from point $q$.

**Step 1:** Partition $Q$ into four parts $Q_1, Q_2, Q_3$, and $Q_4$ by extending horizontal and vertical lines through $q$. Let $Q_1$ be the NE-part.

**Step 2:** Partition $Q_1$ into rectangles by extending the vertical edges of $Q_1$.

**Step 3:** $R_1 = \phi$; $Queue = \phi$; Add to Queue the rectangle of $Q_1$ with $q$ as its vertex;

**Step 4:** While the Queue is not empty do

i. Delete a rectangle *curr* from the Queue;

ii. Add to Queue all E-neighbors of *curr*;

iii. $R_1 = R_1 \cup curr$;

**Step 5:** By following a method similar to Steps 2-4, compute the x-monotone visibility polygons $R_2$, $R_3$, and $R_4$ from $q$ for $Q_2$, $Q_3$, and $Q_4$, respectively.

**Step 6:** Let $R = R_1 \cup R_2 \cup R_3 \cup R_4$;

**Step 7:** Compute y-monotone visibility polygon for $R$ from $q$ by performing computations similar to Steps 1-6, which gives the required solution.

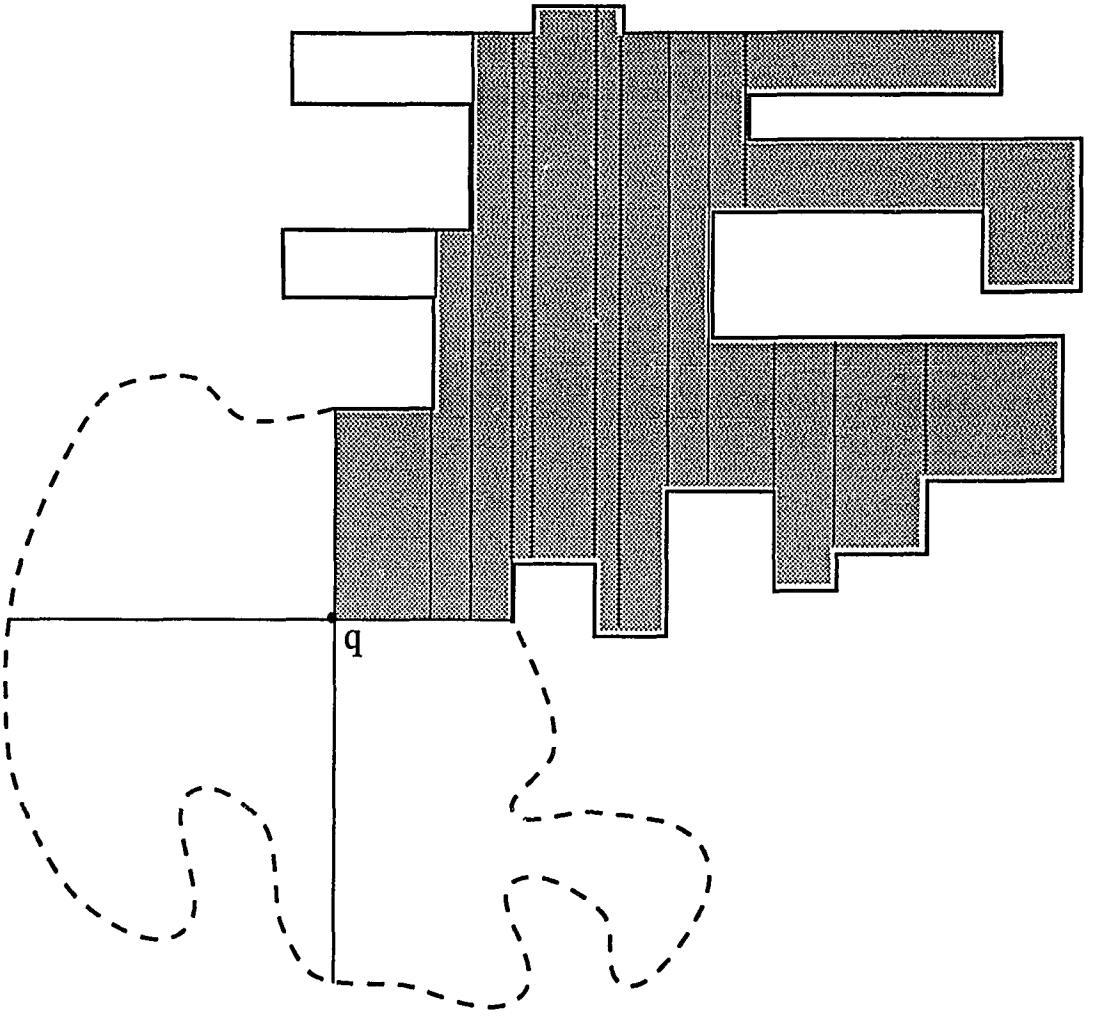



Figure 2.4: Illustrating the x-monotone visibility polygon for the NW-part of polygon P.

**Lemma 2.3:** *The S-Visibility Polygon Algorithm correctly computes the s-visibility polygon from point q.*

**Proof:** Step 4 correctly computes the x-monotone polygon for polygon $Q_1$ from point $q$ as follows. The partial solution $R_1$ remains an x-monotone visibility polygon throughout the iteration of the while loop since only rectangles lying to the east of the boundary of the partial solution are added. Thus rectangles to construct the solution $R_1$ are those rectangles that can be connected by an x-monotone path from $q$. To complete the argument, we need to establish that all rectangles that can be connected by an x-monotone path from $q$ are present in $R_1$ when the loop terminates. Assume to the contrary that there exists a rectangle r' that can be connected from $q$ by an x-monotone path that is not present in $R_1$ when the while loop terminates. Then consider the x-monotone path connecting $q$ to $r$. The path stabs a sequence of vertical rectangles $r_1, r_{i_1}, r_{i_2}, ..., r_{i_p}, r'$. Since $q$ is the SW vertex of $Q_1$, each rectangle in the above sequence lies to the east of its predecessor. This implies that at some point during the iteration of the while loop, the E-neighbor of a rectangle lying on the boundary of the partial solution was not added to the solution, a contradiction to the while loop condition. Symmetrical arguments establish the correctness of steps 5 and 6. □

**Theorem 2.1:** *The s-visibility polygon from a point q inside a simple orthogonal polygon P can be computed in $O(n)$ time.*

**Proof:** This theorem can be proven by examining the algorithm's component steps. Step 1 can be done in $O(n)$ time by simply checking the intersections of vertical and horizontal lines through $q$ with the edges of the polygon. Step 2 can be done by trapizoidalizing the polygon in $O(n)$ time [12]. Step 3 can be done in constant time. Each rectangle enters the queue at most once and, during each iteration, at

least one rectangle is dequeued and hence Steps 4 and 5 take $O(n)$ time each. Step 6 can be done in constant time. Similarly, Step 7 can be done in $O(n)$ time. Therefore, the total time adds up to $O(n)$. □

It would be interesting to find a point inside the polygon from which the total area of s-visibility is maximized. This problem can be approached as follows. We partition the polygon into rectangular cells by extending the edges of the polygon into its interior. Now observe that the s-visibility polygons from points inside a rectangular cell are identical. Hence we can compute the s-visibility polygon from a point in each rectangular cell and report the one having the maximum s-visibility area. Since there can be potentially $\Omega(n^2)$ rectangular cells, the time complexity of the algorithm developed by following the above approach is $O(n^3)$.

# Chapter Three

## Illuminating Simple Polygons by Flood-Lights

### 3.1 Introduction

Several interesting problems arise when we attempt to illuminate the interior of a polygon by placing point light sources. One such problem is the placement of a minimum number of point sources to illuminate the interior of a simple polygon completely. Recall that a point $p$ inside a polygon $Q$ is said to be illuminated by a point $q$ (i.e., $p$ and $q$ are visible to each other) if the line segment connecting $p$ and $q$ does not intersect with the exterior of the polygon. In most illumination problems, visibility from a point is allowed in all directions ($360^0$ angular aperture). Recently, several researchers have considered illumination problems that restrict visibility to within a certain angular aperture [13,14,15]. A light source whose illumination angle is restricted to $\alpha$-degrees is called an $\alpha$-flood-light. It has been established that an orthogonal polygon of $n$ sides can always be illuminated by placing $\lfloor 3(n-1)/8 \rfloor$ $90^0$-flood-lights at its vertices [14], which leads to a simple linear time placement algorithm. Illuminating non-orthogonal polygons by flood-lights seems to be a harder problem. It has been established that a simple polygon is not necessarily illuminated even if all of its vertices contain $90^0$-flood-lights [15]. An important open problem is the determination of the minimum number of $90^0$-flood-lights required to illuminate a simple polygon [14].

In this chapter, an approach based on a construction using beam-machines is used to show that the problem of illuminating a simple polygon by the minimum number of $90^0$-flood-lights is NP-complete.

### 3.2. The Reduction

Consider a simple polygon of $n$ sides represented by a list of its vertices in the order of the clockwise traversal of its boundary. Our goal is to determine the complexity of illuminating the interior of a simple polygon by $90^0$-flood-lights.

**Assumptions:** (i) Flood-light placement is only at the vertices. (ii) Unless indicated otherwise, the default angular aperture of a flood-light is taken as $90^0$. (iii) The aperture of a flood-light can be adjusted to any angle less than $90^0$, but not greater.

**The Flood-light Illumination Problem (FIP)**

**Instance:** A simple polygon $P$ of $n$ sides, a positive integer $m$.

**Question:** Can $P$ be illuminated by at most $m$ flood-lights?

**The Satisfiability Problem (SAT)**

**Instance:** A collection $W = \{C_1, C_2, C_3, ..., C_k\}$ of clauses of a finite set of boolean variables $X = \{x_1, x_2, x_3, ..., x_r\}$; variable $x_i$ occurs $k_{i_1}$ times as $x_i$ and $k_{i_2}$ times as $\overline{x_i}$.

**Question:** Is there a truth assignment for variables in $X$ that satisfies all the clauses?

Given an instance $I_1$ of SAT, we convert it in polynomial time to an instance $I_2 = (P, m)$ of FIP such that $I_1$ is satisfiable if and only if the interior of $P$ can be illuminated by $m$ or fewer flood-lights, where $m$ can be expressed in terms of the number of literals and the number of variables in $I_1$. Our reduction is similar to the construction developed by Culberson and Reckhow [17], where an approach based on beam-machines is used to reduce the satisfiability problem to the problem of covering a simple polygon by the minimum number of convex polygons.

**Beam Machine:** One key structure used in the construction of $P$ is the beam-machine. A beam-machine consists of a seven sided simple polygonal chain. We label the vertices of a generic beam-machine clockwise as $a, b, c, d, d', c', b'$, and $a'$ (see Figure 3.1a). The opening between $a$ and $a'$ is referred to as the mouth. For the purpose of exposition we can think the interior of a beam-machine as the interior of a simple polygon formed by joining the vertices $a$ an $a'$. The magnitudes of the interior angles of the vertices of a beam-machine are indicated in Table 1.

**Table 1**

| Vertices | Magnitude of Internal Angle |
|---|---|
| b, b' | between $180^0$ and $270^0$ |
| a, a', c, c' | between $90^0$ and $180^0$ |
| d, d' | exactly $90^0$ |

**Property 3.1:** *All points in the interior of the beam-machine are visible from vertex d.*

**Property 3.2:** *All points in the interior of the beam-machine are visible from vertex d'.*

**Property 3.3:** *The vertices c and c' are not visible from the mouth of the beam-machine.*

There are two distinct placements to illuminate a beam-machine by one flood-light: one is the placement at vertex $d$ **(placement-d)** and the other is at vertex $d'$ **(placement-d')**

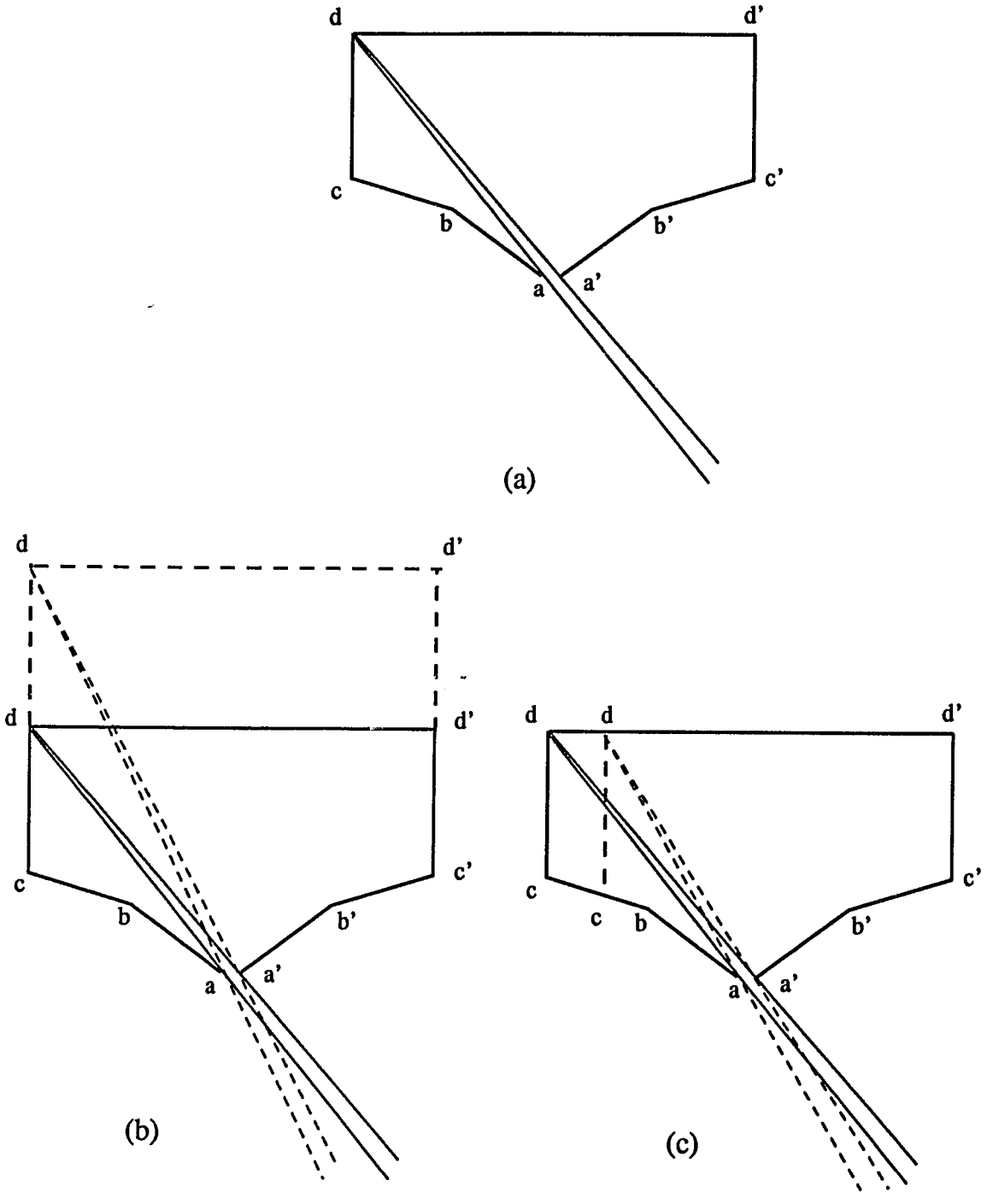
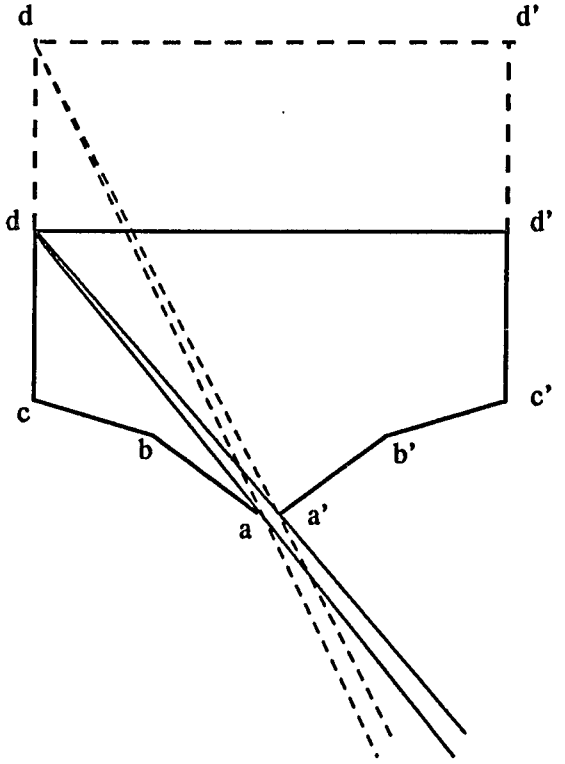
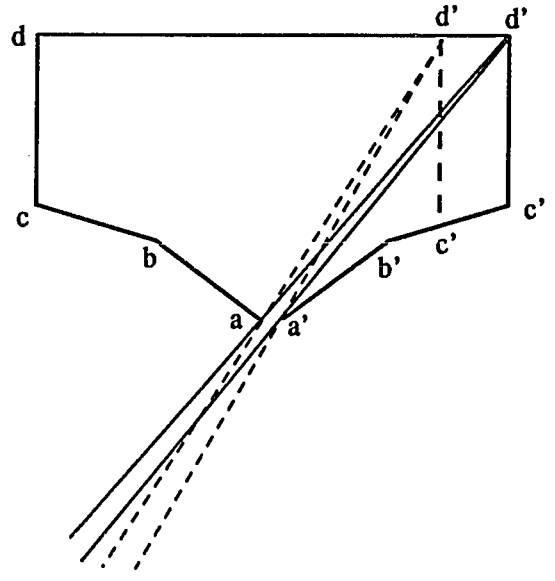



Figure 3.1: Illustrating the construction of the beam machine




(a): Adjusting beam orientation for placement-d




(b): Adjusting beam orientation for placement-d'

Figure 3.2: Illustrating the adjustment of beam orientations for placement-d and placement-d'

**Lemma 3.1:** *One flood-light is necessary and sufficient to illuminate a beam-machine.*

**Observation 3.1:** (Beam Formation) Consider the illumination of a beam-machine by using placement-d or placement-d'. Both placements create a thin illumination beam that extend to the exterior of the beam-machine, in addition to the complete illumination of its interior. The thin beam corresponding to placement-d is inclined to the right and that corresponding to placement-d' is inclined to the left. These two ways of illuminating a beam-machine and the creation of a thin beam can be used to simulate an illumination switch. Placement-d can be viewed as setting the beam-machine to the "on" state and placement-d' as setting it to the "off" state

**Beam Adjustment:** In our construction, we need to use beam-machines that can generate illumination beams whose width and orientation can be adjusted. The width of the beam can be adjusted by altering the opening of the mouth of the beam-machine. The orientation of the beam can be changed by adjusting the height or the width of the beam-machine as illustrated in Figure 3.1b and Figure 3.1c, respectively. Of course, we need to be careful in altering the length of the edges so that the visibility properties (Property 3.1, Property 3.2, and Property 3.3) of the resulting beam-machine do not change.

**Observation 3.2:** (Beam Adjustment) It should be noted that the orientation of the beams extending from the mouth of the beam-machine from placement-d and placement-d' are dependent if one adjust the height of the beam-machine. Additionally, they are independent if one adjust the width by extending or contracting one side. Unlike the first method, the latter method will result in an asymmetrical beam-machine with independent beams.

**Observation 3.3:** (Beam Adjustment) It would be convenient for us to be able to change the orientation of each beam produced by a beam-machine in a systematic way. This can be achieved by using only a particular type of adjustment for placement-d and the other for placement-d'. For example, we could use only height-adjustment to change the orientation of the beams produced by placement-d and only width-adjustment to change the orientation of the beam produced by placement-d' (see Figures 3.2a-b).

**The Background Variable Structure:** The background variable structure **(BVS)** is a fourteen sided polygonal chain whose vertices are labeled as $v_1, v_2, v_3, ..., v_{15}$. The interior angles at $v_3, v_7, v_9$ and $v_{13}$ are each required to be $90^0$ or less (see Figure 3.3a). The interior of a BVS is taken as the interior of the polygon formed by joining vertices $v_1$ and $v_{15}$. We can think of a BVS as consisting of two symmetrical wings (the left wing and the right wing). The BVS satisfies the following visibility properties:

**Property 3.4:** *The interior of a BVS can be completely illuminated by placing floodlights at vertices $v_7$, $v_9$ and either $v_3$ or $v_{13}$.*

**Property 3.5:** *The extension of segment $\overline{v_3, v_4}$ into the interior of the polygon meets the segment $\overline{v_{11}, v_{12}}$ in its interior. Similarly, the extension of segment $\overline{v_{12}, v_{13}}$ meets segment $\overline{v_4, v_5}$ in its interior.*

**Property 3.6:** *Vertices $v_3$ and $v_{13}$ are visible to each other.*

**Variable Generator:** We construct a variable generator corresponding to variable $x_i$ by gluing beam-machines and spikes onto the boundary of a BVS. (Recall that variable $x_i$ occurs $k_{i_1}$ times as $x_i$ and $k_{i_2}$ times as $\overline{x_i}$). We add $k_{i_1}$ beam-machines $b_1, b_2, b_3, ..., b_{k_{i_1}}$ on the top of the right wing (side $\overline{v_8, v_9}$) and $k_{i_2}$ beam-machines $d_1, d_2, d_3, ..., d_{k_{i_2}}$ on the top of the left wing (side $\overline{v_7, v_8}$). Corresponding to the $k_{i_1}$ beam-machines, we attach $k_{i_1}$ spikes $s_1, s_2, s_3, ..., s_{k_{i_1}}$ to the bottom of the right wing (side $\overline{v_{11}, v_{12}}$). Similarly, we attach $k_{i_2}$ spikes $q_1, q_2, q_3, ..., q_{k_{i_2}}$ to the bottom of the left wing (side $\overline{v_4, v_5}$)

**Property 3.7:** *Spikes $s_1, s_2, s_3, ..., s_{k_{i_1}}$ can be illuminated either by setting beam-machines $b_1, b_2, b_3, ..., b_{k_{i_1}}$ to the "on" state or by placing a $90^0$-flood-light $v_3$.*

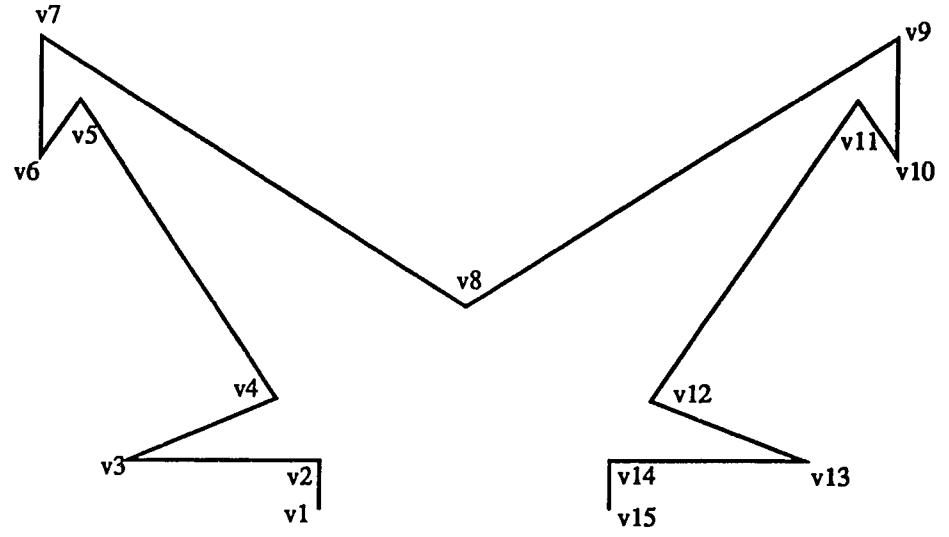
**Property 3.8:** *Spikes $q_1, q_2, q_3, ..., q_{k_{i_2}}$ can be illuminated either by setting beam-machines $d_1, d_2, d_3, ..., d_{k_{i_2}}$ to the "off" state or by placing a $90^0$-flood-light at vertex $v_{13}$.*

**Illuminating Variable Generators:** Consider a variable generator $V_i$ corresponding to variable $x_i$. The background of $V_i$ can be illuminated by placing three flood-lights: two at vertices $v_7$ and $v_9$, and one at either $v_3$ or $v_{13}$. Each beam-machine can be illuminated either by placement-d or by placement-d' .The spikes can be illuminated either by the beams coming from the beam-machines or by the flood-light placed at either $v_3$ or $v_{13}$. If $x_i$ is set "false" then the beam-machines are illuminated by using placement-d' and the beams originating from the beam-machines on the left wing illuminate the spikes on the left wing; the spikes on the right wing are illuminated by placing a light at $v_3$. The beams originating from the beam-machines in the right wing escape away from the mouth of $V_i$. Figure 3.3b shows the illumination

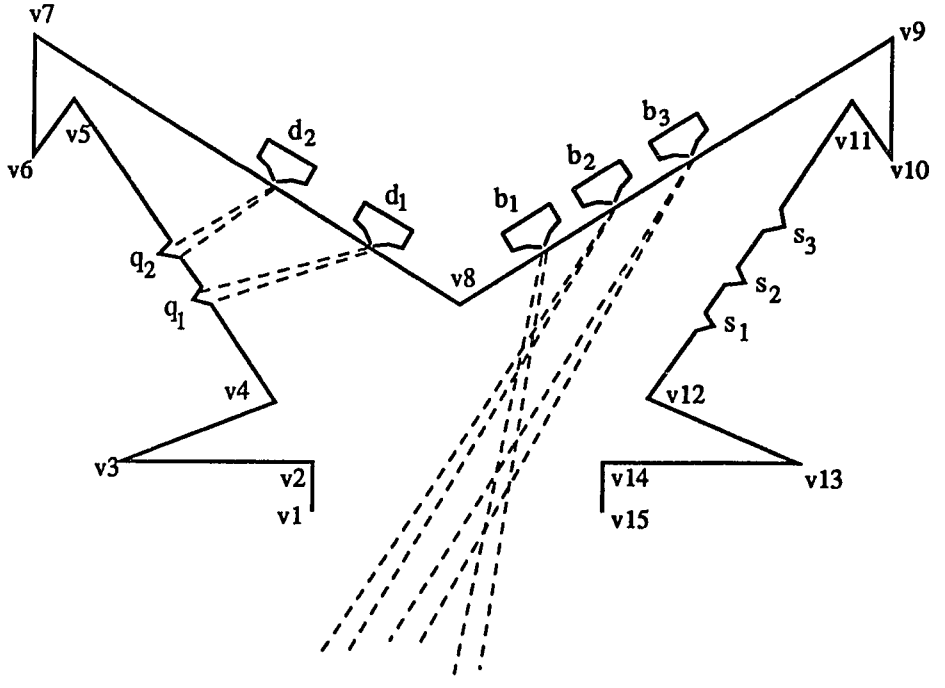
when $x_i$ is set "false". On the other hand, if $x_i$ is set "true" then the beam-machines are illuminated by using placement-d and the beams originating from the right wing illuminate the spikes on the right; the spikes in the left wing are illuminated by the flood-light at $v_{13}$. The beams originating from the left wing escape away from the mouth of $V_i$.

**Lemma 3.2:** *The variable generator $V_i$ can be illuminated by $(k_{i_1}+k_{i_2})+3$ flood-lights.*

**Polygon Construction:** We construct a simple polygon by adding clause checkers and variable generators to the sides of a long rectangle $R$. Let $a, b, c$, and $d$ be the corners of $R$. We add $k$ clause checkers at the bottom side of $R$. Each clause-checker is a small triangular spike. No clause checker can be fully illuminated from the four vertices of $R$. For each variable $x_i$, we construct a variable generator containing $k_{i_1} + k_{i_2}$ beam-machines. These $r$ variable generators are attached to the top side of $R$ (see Figure 3.3). Finally, we add a spike $z$ in the south-east corner of $R$; the spike $z$ is such that it can be illuminated from vertex $a$ but not from any vertex lying on or above the edge $\overline{b,c}$. The beam-machines of each variable generator can be adjusted so that the beams escaping from the mouth of the variable generators illuminate the corresponding clause checker. Let $P$ denote the resulting simple polygon. It is easy to see that $P$ can be constructed in polynomial time.




(a)




(b)

Figure 3.3: Illustrating the construction of a variable generator

**Lemma 3.3:** *All clauses in $W$ are satisfiable if and only if $P$ can be illuminated by $m$ or fewer flood-lights, where $m$ can be expressed in terms of the number of literals and the number of variables.*

**Proof:** (only if part) Assume that there exists a truth assignment that satisfies all clauses in $W$. If variable $x_i$ is assigned "true", then we illuminate the beam-machine associated with the corresponding generator by using placement-d. By Lemma 3.2, $g = \sum_{i=1}^{r}((k_{i_1} + k_{i_2}) + 3)$ flood-lights can illuminate the interior of all variable generators. Since $E$ is satisfiable, all clauses evaluate "true", and hence, each clause checker spike is illuminated by at least one beam-machine. The remaining area of the polygon is illuminated by the flood-light placed at vertex $a$. Hence $P$ is illuminated by $m = g + 1$ flood-lights.

(if part) Suppose that the polygon can be illuminated by $m = g+1$ $90^0$-flood-lights. The variable generator corresponding to the variable $x_i$ needs at least $(k_{i_1} + k_{i_2}) + 3$ $90^0$-flood-lights to be illuminated. Hence, $g = \sum_{i=1}^{r}((k_{i_1} + k_{i_2}) + 3)$ $90^0$-flood-lights are needed to illuminate all variable generators. The only way to illuminate clause checkers is by the beams coming from the variable generators. If a clause checker corresponding to clause $c_j$ is illuminated by a beam coming from the beam machine on the left wing of the variable generator corresponding to variable $x_q$, then this variable must occur uncomplemented in clause $c_j$; hence we set $x_q$ "true" to evaluate $c_j$ "true". Similarly, if the beam was coming from the right wing, then we would set $x_q$ "false" to evaluate $c_j$ "true". Since each clause checker is illuminated, all clauses can be evaluated "true" by the above assignment process.

It is easily seen that FIP belongs to the class NP. Hence we have the following theorem.

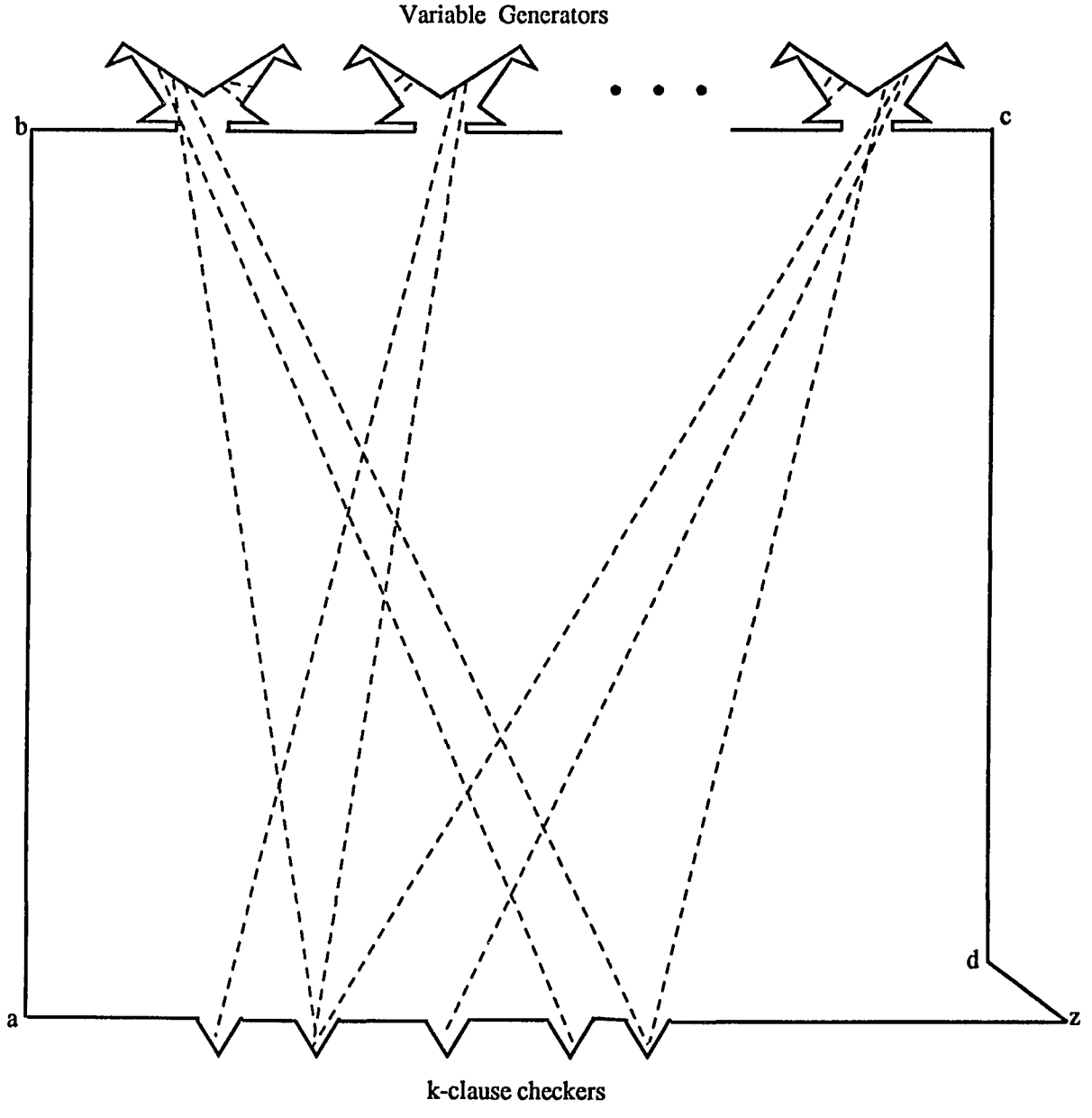



Figure 3.4: Illustrating the final construction

**Theorem 3.1:** *The* $90^0$ *flood-light illumination problem (FIP) is NP-Complete.*

It may be noted that the construction of beam-machines, variable generators, and the final polygon remain identical even if flood-lights are required to be flush with the edges of the polygon. Hence we have the following corollary.

**Corollary 3.1:** *The* $90^0$*-flood-light illumination problem is NP-Complete even if all flood-lights are required to be flush with the edges of the polygon.*

# Chapter Four

## Conclusion and Extensions

We presented a linear time algorithm for computing the s-visibility polygon from a point inside a simple orthogonal polygon. This algorithm can be modified to determine the s-visibility polygon from a point inside any simple polygon (not necessarily orthogonal) without increasing its time complexity. The algorithm is optimal within a constant factor.

We can extend the notion of s-visibility to obtain k-visibility as follows. Two points inside a polygon are said to be **k-visible** if they can be connected by a simple path of $k$ horizontal or vertical line segments that do not intersect with the exterior of the polygon. It would be interesting to explore the visibility properties of polygons under k-visibility. We have made some progress in developing an algorithm to compute the k-visibility polygon from a given point inside a polygon. This result will be reported in the future.

We proved that the flood-light illumination problem (FIP) is NP-complete when the angular aperture of the flood-light is restricted to $90^0$. A simple inspection of beam machines, variable generators, and the constructed polygon reveals that the problem remains NP-complete even if flood-lights are required to be flush with polygon edges. A natural generalization would be to determine the complexity of FIP when angular aperture of flood-lights is some $\alpha$ in the range $0 < \alpha \leq 360^0$. We have been partially successful in making this generalization. The main components for this generalization can be sketched as follows.

The beam-machine used for an $\alpha$-flood-light consists of a seventeen-sided polygonal chain as shown in Figure 4.1. It is convenient to view the beam-machine as consisting of three distinguishable components: (i) the **left wing** ($defghi$), (ii) the **right-wing** ($d'e'f'g'h'i'$), and (iii) the **body** (abcdii'd'c'b'a'). The beam-machime satisfies the following structural and visibility properties.

**Property 4.1:** *The interior angle at vertex e and angle did' are at most $\alpha$ each. Similarly, the interior angle at vertex e' and angle d'i'd are at most $\alpha$ each.*

**Property 4.2:** *Vertices $f, g, h, i$, and $i'$ are visible from vertex e. Similarly, vertices $f', g', h', i'$ and $i$ are visible from vertex e'.*

**Property 4.3:** *Vertices $d, c, b, a, a', b', c'$ and $d'$ are visible from both vertices $i$ and $i'$.*

**Property 4.4:** *Vertex c is not visible from vertex d'. Similarly, vertex c' is not visible from vertex d.*

**Lemma 4.1:** *Three $\alpha$-flood-lights are necessary and sufficient to illuminate the beam-machine.*

**Proof:** (sufficiency) Two $\alpha$-flood-lights, one at $e$ and the other at $e'$, can illuminate the left-wing and right-wing, respectively (Property 4.2). The remaining part, the body of the beam-machine, can be illuminated by a third light at vertex $i$ or $i'$ (Property 4.3).

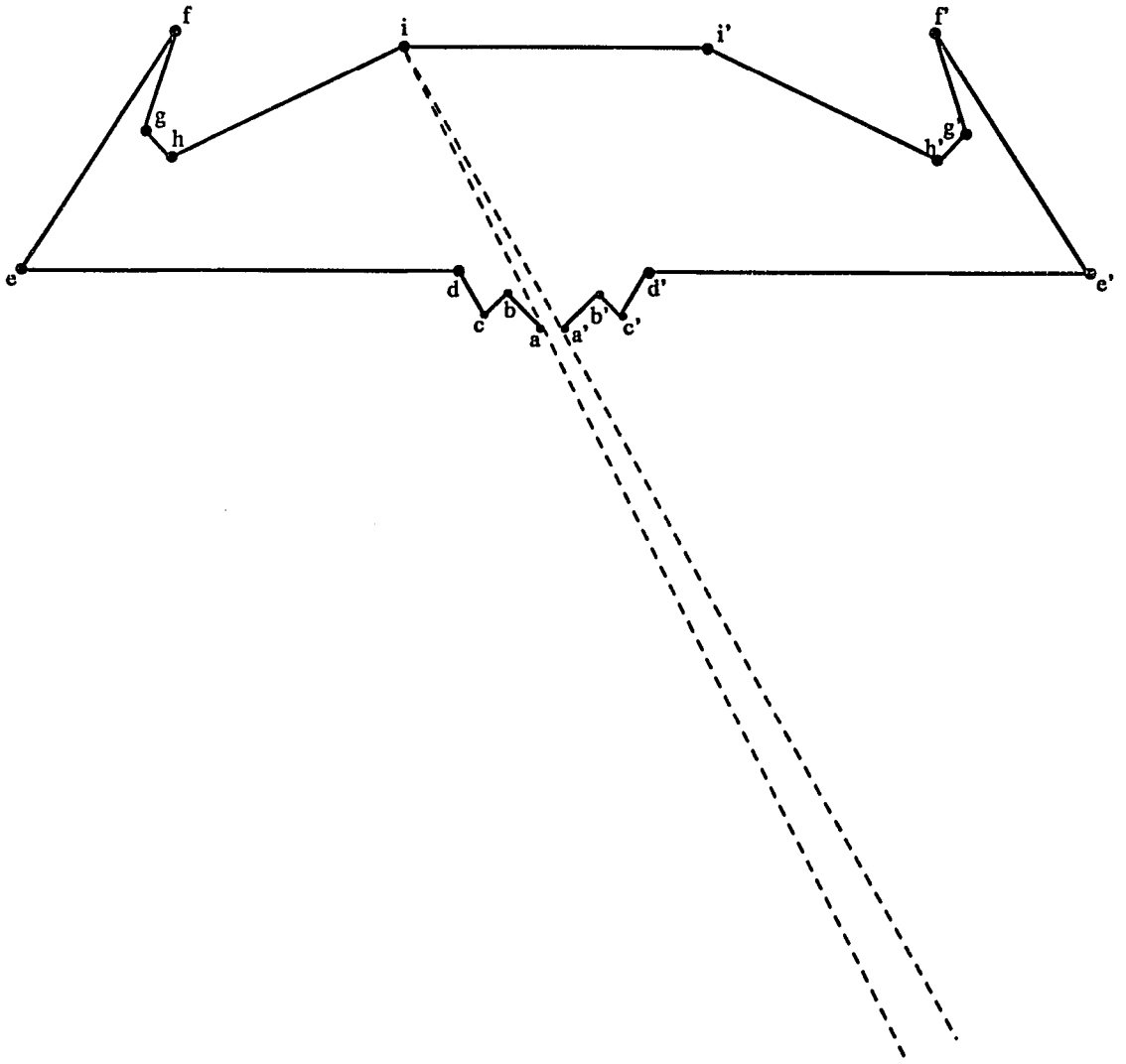



Figure 4.1: Illustrating a beam-machine for an $\alpha$–flood-light

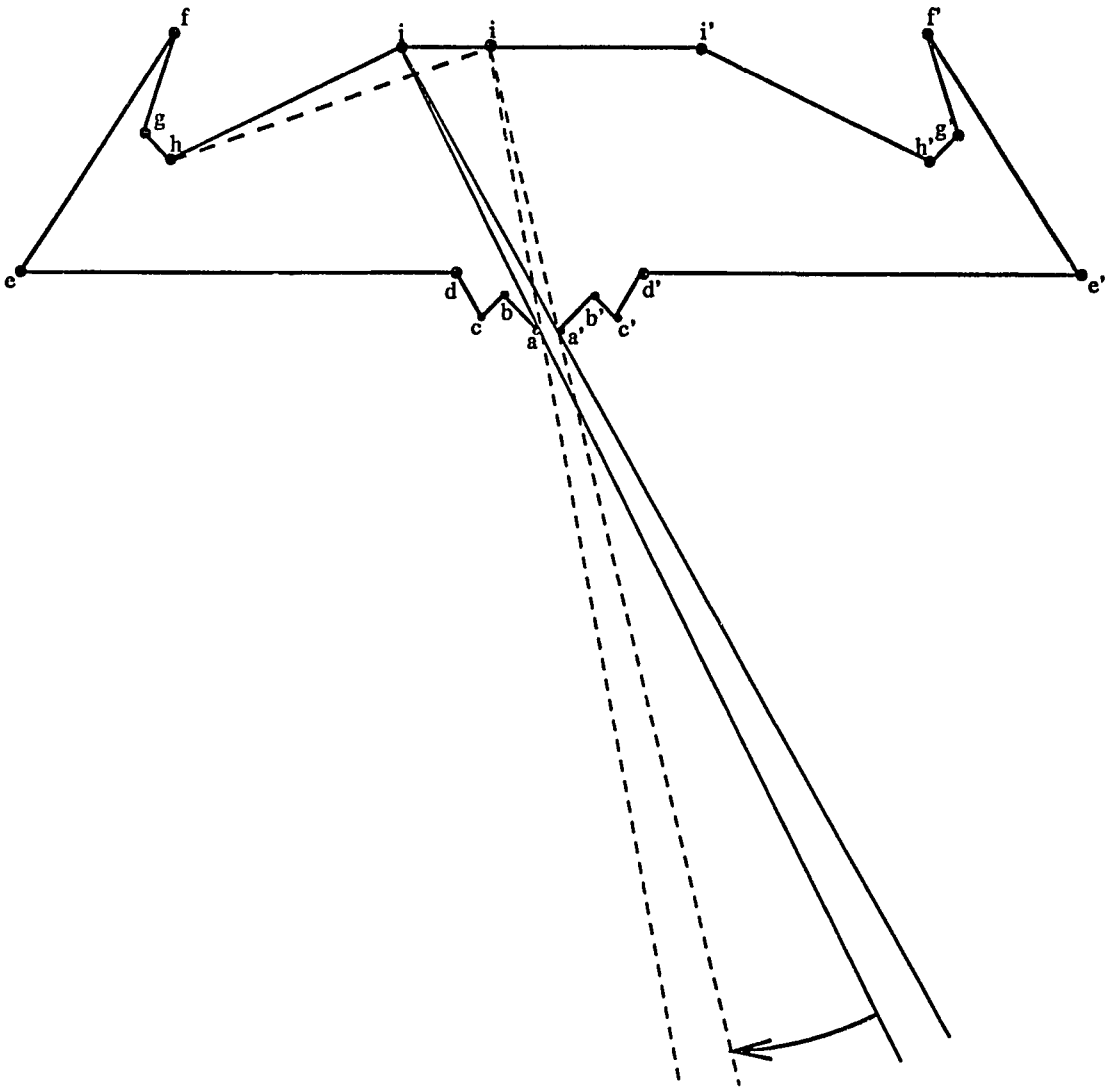



Figure 4.2: Illustrating beam rotation

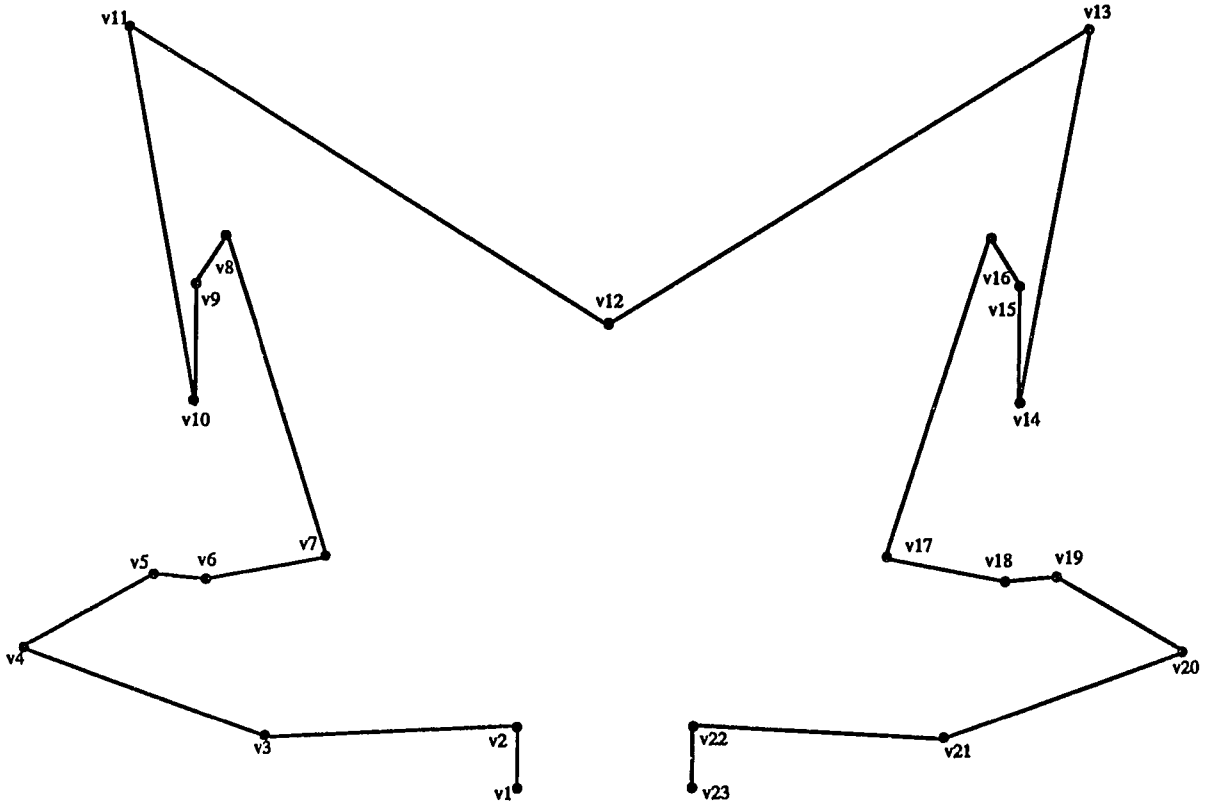



Figure 4.3: Illustrating the construction of a background variable structure

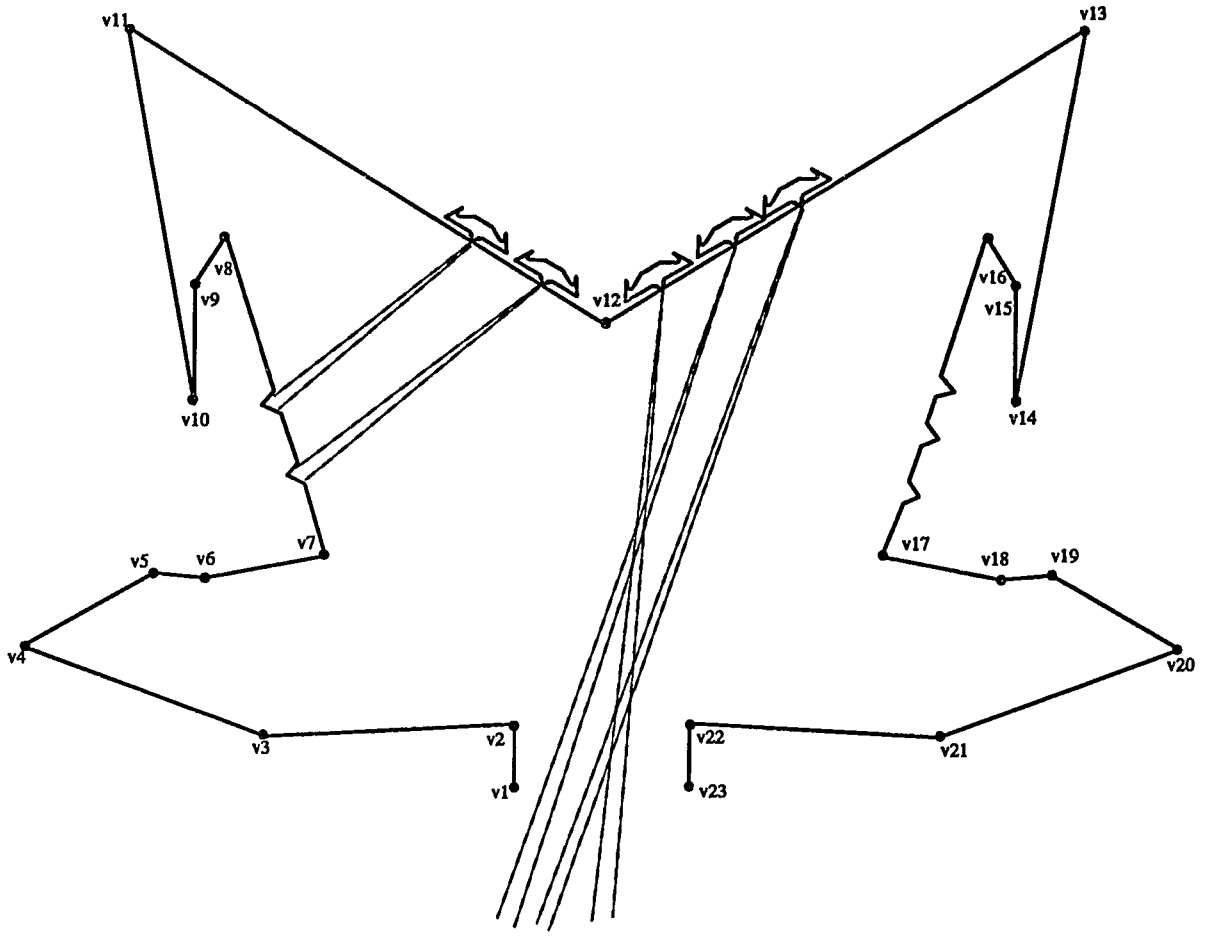



Figure4.4: Illustrating the variable generator

(necessity) Since chain $i, h, g, f$ is concave and since the extension of edge $\overline{(c,d)}$ meets edge $\overline{(h,i)}$, the chain can be illuminated only from vertex $e$. Similarly, concave chain $i', h', g', f'$ can be illuminated only from the vertex $e'$. Hence, two $\alpha$-flood-lights are needed to illuminate the wings of the beam-machine; and the structure forces them to be placed at $e$ and $e'$. The third $\alpha$-flood-light is needed to illuminate the body. □

The beam-machine can thus be illuminated by placing three $\alpha$-flood-lights in two distinct ways: one is the placement at vertices $e, e'$, and $i$ (**placement-i**) and the other is the placement at vertices $e, e'$ and $i'$ (**placement-i'**). Both placements result in the formation of a thin beam extending away from the opening of the beam-machine (see Figure 4.1). By following the same convention as in Chapter Three, placement-i and placement-i' can be viewed as setting the beam machine to the "on" and the "off" state, respectively.

The orientation of the right beam can be adjusted by moving vertex $i$ as illustrated in Figure 4.2. Of course, all visibility properties stated above should remain unaltered by the adjustment. A straightforward inspection of the beam-machine shows that this can be done easily. Figure 4.3 and 4.4 demonstrate the modified background variable structure (BVS) and the variable generator, respectively. The final polygon can be constructed in a similar way as in Figure 3.4. The details are currently in progress.

## References


[1 ] J. O'Rourke, "Art Gallery Theorems and Algorithms," Oxford University Press, 1987.

[2 ] T.C. Shermer, "Recent Results in Art Galleries," *Proceedings of the IEEE*, 1992, pp. 1384-1399.

[3 ] D. T. Lee, "Visibility of a Simple Polygon," *Computer Vision, Graphics, And Image Processing*, Vol. 22, 1983, pp. 207-221.

[4 ] H. ElGindy and D. Avis, "A Linear Time Algorithm for Computing the Visibility Polygon From a Point," *Journal of Algorithms*, Vol. 2, 1981, pp. 186-197.

[5 ] S. Ntafos and M. Tsoukalas, "Optimal Placement of Guards," *Proceedings of the Third Canadian Conference on Computational Geometry*, 1993, pp. 122-125.

[6 ] R. Motwani, A. Raghunathan and H. Saran, "Covering Orthogonal Polygons With Star Polygons: The Perfect Graph Approach," *Journal of Computer and Systems Sciences*, Vol. 40, 1990, pp. 19-48.

[7 ] R. Reckhow and J. Culberson, "Covering Simple Orthogonal Polygons With a Minimum Number of Orthogonally Convex Polygons," *Proceedings of the Third Annual Symposium on Computational Geometry*, 1987, pp. 268-277.

[8 ] R. Motwani, A. Raghunathan and H. Saran, "Perfect Graphs and Orthogonally Covers," *SIAM J Diacrete Math*, 1989, pp. 371-392.

[9 ] L. Gewali, M. Keil, and S. Ntafos, "On Covering Orthogonal Polygons With Star- Shaped Polygons", *Information Sciences*, Vol 65, 1992, pp. 45-63.

[10 ] D. Avis and G.T. Toussaint, "An Optimal Algorithm for Determining the Visibility of a Polygon from an Edge," *IEEE Trans. Comput. C-30 (1981b)*, 1981, pp. 910-914.

[11 ] S. Ntafos, "On Gallery Watchman in Grids," *Info. Proc. Let. 23*, 1986, pp. 99-102.

[12 ] B. Chazelle, "Triangulating a Simple Polygon in Linear Time," *Proc. 31st Annual Symp. on Foundation of Computer Science*, 1990, pp. 220-230.

[13 ] J. Czyzowicz, E. Rivera-Campo, and J. Urrutia, "Optimal Flood-light Illumination of Stages," *Proceedings of the Fifth Canadian Conference on Computational Geometry*, 1993, pp. 393-398.

[14 ] V. Estivill-Castro and Jorge Urrutia, "Optimal Flood-light Illumination of Orthogonal Art Galleries," *Proceedings of the Sixth Canadian Conference on Computational Geometry*, 1994, pp. 81-86.

[15 ] J. O'Rourke and D. Xu, "Illumination of Polygons with $90^0$ Vertex Lights," *Snapshots of Computational and Discrete Geometry*, Vol. 3, Editors: David Avis and P. Bose, McGill University, 1994.

[16 ] M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan, "Triangulating a Simple Polygon," *Info. Proc. Let. 7*, 1978, pp. 175-179.

[17 ] J.C. Culberson and Robert A. Reckhow, "Covering Polygons is Hard," *Journal of Algorithms* Vol. 17, 1994, pp. 2-44.