Université d'Ottawa • University of Ottawa

Cutting polygons and a problem on illumination of stages

by

Felipe Contreras

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Master in Computer Science

School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

The Master in Computer Science program is a joint program with
Carleton University, administered by the Ottawa-Carleton
Institute for Computer Science

Your file Votre référence
ISBN: 0-612-90049-5
Our file Notre référence
ISBN: 0-612-90049-5

Abstract

This work presents the solution to two problems in Computational Geometry. First, we introduce an algorithm to calculate (provided an $O(n \log n)$ preprocessing or linear if the polygon is convex) the area of an n-gon "cut" by a query interior segment in $O(\log n)$ time. As an application we also show how to find the line cutting $\frac{1}{\rho}$ of the area of a convex polygon and parallel to a given line. Secondly, we show how to illuminate a stage represented by a line segment s, with floodlights placed at n points above s such that the sum of their angles is minimized. The algorithm runs in $\Theta(n \log n)$ time and we include a videotape presenting it.

Acknowledgments

I want to thank: Jorge Urrutia for his patience and support in supervising this work, to Harvinder Singh for proofreading it, to Margaret Urrutia for making the final reading and the friends and professors from UOttawa, Instituto de Matemáticas and Facultad de Ciencias UNAM México for their encouragement. Special thanks to my friends Liz and Victor for all their understanding. Thanks also to J. Czyzowicz for suggesting the idea of our video and his enthusiasm (beating the deadline through the winter storm!) and to Alvaro, Francisco and Gina for their unconditional support.

The author was sponsored by CONACyT México scholarship:
110147/110159

Contents

List of Figures

Chapter 1

Introduction

This work deals with two problems in Computational Geometry.

The first problem consists of calculating the area of a polygon cut by a segment, where the segment divides the polygon into only two subpolygons.

Our approach uses a *preprocessing time*; that is, we obtain and store some useful information about the structure of the polygon at the beginning of the algorithm. Afterwards, i.e. at *processing time*, we may have several *query cuts*. This means that the preprocessing time translates into time saved in processing. The traditional method consists in calculating the area of the cut part as another polygon, blindly ignoring information gained from the previous cuts.

We first give an introduction on calculating areas and volumes of arbitrary polygons and polyhedra (Chapter 2). Next, we analyze the problem of determining the area of a convex polygon cut by a line, and give an application which consists of determining the line bisecting the area of a convex polygon (Chapter 3). Finally, we develop the general case of cutting simple

polygons, followed by a note on cutting polygons and polyhedra using lines and planes (Chapter 4).

Our second problem is the optimal floodlight illumination of stages. The stage is represented by a line segment and the floodlights originate from a set of points above the segment. The condition required is to minimize the sum of the apertures of the floodlights (Chapter 5).

In this chapter we also describe the solution given in [CRCU93], with an improvement on the complexity for the case when the points are vertices of a convex polygon (cf. Problem 5.3.1). It is an introduction to the companion videotape of this work.

1.1 Related work

1.1.1 Areas

The problem of calculating areas has been studied since the times of the ancient Babylonians, whose triangulation algorithm is stated in modern terms in Algorithm 2.1.1. The Greeks were known to have measured polygonal land areas (hence the origin of the word "geometry") developing a variety of formulas for special cases of polygons and volumes of polyhedra (e.g. regular, truncated, etc.).

The methods used to calculate areas range from decomposing the polygons/polyhedra into simpler regions, for example with parallel lines/planes or using triangulations, up to calculus methods. Here we will show one of the methods commonly used by present-day programmers.

We can calculate the area $\mathcal{A}(P_n)$ of the polygon P_n with vertices $\{p_1 =$

$p_{n+1}, p_2, \ldots, p_n\}$, using the following formula:

$$\mathcal{A}(P_n) = \frac{1}{2}\sum_{i=1}^{n}[x_i y_{i+1} - x_{i+1} y_i] \tag{1.1}$$

where $p_i = (x_i, y_i)$. Equation 1.1 (known as the *parallelogram formula* or *polar formula*) can be derived from Green's theorem and is equivalent to the following:

$$\mathcal{A}(P_n) = \frac{1}{2}\sum_{i=1}^{n}\begin{vmatrix} x_i & y_i \\ x_{i+1} & y_{i+1} \end{vmatrix}$$

which is the sum of the half-areas of the quadrilaterals defined by the vertices of P_n seen as vectors from the origin. The geometrical equivalent of this formula is shown in Section 2.2.

Another version of this formula is:

$$\mathcal{A}(P_n) = \frac{1}{2}\sum_{i=2}^{n} x_i(y_{i+1} - y_{i-1}), \tag{1.2}$$

known as the *surveyor's formula*, using the analogy to a surveyor traveling around the vertices of P_n (see an interesting paper describing both polar and surveyor formulas in [Str93]).

In any case, the importance of this solution for the area of a polygon consists in its simple generalization to $\boldsymbol{R}^3$ (see Section 2.3 and [Lig88][All86]).

A different approach for calculating areas is given by Pick's theorem (see [Ste69][GS93][DR95]). It describes the area of a polygon whose vertices have integer coordinates in terms of the *lattice points* (integer pairs) inside it. *Pick's formula* is:

$$\mathcal{A}(P_n) = I + B/2 - 1, \tag{1.3}$$

where $I =$"lattice points interior to P_n" and $B =$"lattice points on the boundary of P_n".

Yet another approach to calculating areas of polygons is shown in [Rob95], where the author starts with *Heron's formula* for the area of triangles:

$$\mathcal{A}(\Delta abc) = \sqrt{s(s-a)(s-b)(s-c)}$$

where triangle Δabc has edges with lengths a, b and c, and $s = (a+b+c)/2$. Afterwards he observes *Brahmagupta's formula* for cyclical quadrilaterals (a simple polygon is called *cyclical* if we can draw a circle through all of its vertices):

$$\mathcal{A}(abcd) = \sqrt{(s-a)(s-b)(s-c)(s-d)}$$

for a quadrilateral with edges of lengths a, b, c and d and $s = (a+b+c+d)/2$ and tries to generalize them to general cyclical polygons. However the author gives up after finding that even for the heptagon, its formula becomes very complicated to write.

A generalization of *Brahmagupta's formula for general quadrilaterals* is as follows:

$$\mathcal{A}(abcd) = \sqrt{(s-a)(s-b)(s-c)(s-d) - abcd\cos^2\frac{A+C}{2}}$$

where A and C are two non-adjacent angles If the quadrilateral is inscribed in one circle and circumscribed in another then the area is

$$\mathcal{A}(abcd) = \sqrt{abcd}.$$

Finally C. A. Bretschneider (1842) gives another formula:

$$\mathcal{A}(abcd) = \frac{1}{4}\sqrt{4p^2q^2 - (b^2+d^2-a^2-c^2)^2}$$

where p and q are the lengths of the diagonals [CG67].

Skiena [Ski91] calculates the area of a convex polygon cut by a line (or intersecting a half plane) using an algorithm running in linear time.

Some other authors deal with the problem of two overlapping polygons, in order to find the maximum/minimum area of their intersection (see [Ven95]) with a running time of $O(n(n+m))$, or to study the area function $\mathcal{A}(t) = \mathcal{A}(A \cap B + t)$ (see [MSW93]), for the polygons A and B, where $B + t$ is a translation of B by the vector t.

For the calculation of areas and volumes there exists another method very similar to the classical decomposition into parallelograms in the numerical approximation of the integral of a function. In Section 4.8 we call it the *orthogonal projection* procedure. In [All86] and [Spe86] we find a generalization to $\boldsymbol{R}^d$ of the orthogonal and polar methods.

The *ham-sandwich* problem for polygons asks for a single line bisecting simultaneously the area of two linearly separable polygons. Note that there doesn't always exist a solution if the polygons are not separable by a line (one to the left and the other to the right of the line).

Stojmenović in [Sto91] shows a linear time algorithm to solve the ham-sandwich problem for two convex polygons, three convex polyhedra, etc.

Later, Díaz and O'Rourke in [DO90] present an $O(n \log n)$ time algorithm to solve the ham-sandwich problem for any simple pair of linearly separable polygons. In the same paper an orthogonal *foursection* is defined as splitting a simple polygon into four parts with equal area using two perpendicular lines. They present the orthogonal foursection of convex polygons in linear time.

Shermer in [She92] solves the problem of bisecting any simple polygon by a line with a given slope in linear time. He uses a *prune-and-search* procedure to determine the trapezoids to the left, to the right and cut by the bisector, from the elements of a trapezoidization of the polygon.

1.1.2 Illumination

Floodlight illumination problems enjoy increasing interest among the members of the Geometry community.

A *floodlight* is a light source originating from a point called the *apex* and having a specified angle of aperture of at most α. The floodlight is represented by an infinite angular wedge in the plane. A *vertex floodlight* is a floodlight with its apex on a vertex of a polygon. No more than one floodlight is allowed per vertex, unless otherwise specified. A *point floodlight* is a floodlight with an apex on a point in the plane. More than one floodlight is allowed per point, unless otherwise specified. If the aperture angle is 2π, the floodlight is called simply a *light* or a *guard*. A polygon is said to be *illuminated* by a set of floodlights F if it is contained in the union of the wedges of F.

The Fisk proof of Chvátal's theorem shows that a simple n-gon can be illuminated in linear time with at most $\lfloor \frac{n}{3} \rfloor$ vertex lights [Fis78].

A convex polygon can be illuminated with three vertex floodlights whose apertures add up to π [Urr97a].

A convex polygon can also be illuminated with two floodlights if the sum of their apertures is minimized. The algorithm runs in $O(n^2)$ time [ECU95].

An orthogonal polygon, one having its edges parallel to the axes (also

called *traditional art gallery*), can be illuminated in linear time with $\lfloor \frac{n}{4} \rfloor$ vertex floodlights with aperture $\pi/2$. This result has six proofs; the first is in [KKK83] and the sixth in [Urr97b].

There are many other problems on guarding or illuminating polygons with different variations on shapes and kinds of lights, therefore we will refer to the excellent survey on the area [Urr97a].

The stage illumination problem (Problem 5.1.1) was first stated by Urrutia at [BGL+93] and is still open in its general version. In that paper, the authors also point out the following results:

Given $n = k_1 + k_2 + k_3$ and three floodlights with apertures $\alpha_1 + \alpha_2 + \alpha_3 = 2\pi$, we can illuminate k_i points with a floodlight of aperture α_i. From this result, the authors are able to illuminate the whole plane with three floodlights of aperture at most π if their apertures add up to at least 2π. The algorithms for these two results run in $O(n \log n)$ time. Later it was proved that this time is optimal.

Another result [Urr97a] says that four point floodlights with aperture $\pi/2$ suffice to illuminate the whole plane.

An interesting result by O'Rourke et al [OSS95] says that for high values of n, we cannot illuminate a convex n-gon with m floodlights ($m \leq n$) whose apertures add up to π, however nothing is said about low values of n $\{4, 5, \ldots\}$.

1.2 Contributions

We consider that the most important contribution of this thesis is the introduction of preprocessing time in improving area-finding algorithms. It enables us to solve the old problem of finding areas of pieces of polygons and other related problems, "breaching" the optimal linear time barrier.

The main contributions of this work concerning the area problem (Chapters 3 and 4) are a joint research work with Jurek Czyzowicz and Jorge Urrutia. Some of the results here are resumed in [CCAU98]. We state in that paper that the area of any polygon cut by an interior segment can be calculated in logarithmic time with $O(n \log n)$ algorithm (Theorem 4.6.2). The secondary results show that a convex polygons can be preprocesed in linear time to find 1. the area cut by a query line in constant constant/logarithmic time and 2. the bisector of the area with a given direction, in logarithmic time.

Regarding the stage illumination problem (Chapter 5), this work presents a companion video explaining the corresponding algorithm. This video was a joint work with Jurek Czyzowicz, Jorge Urrutia and Eduardo Rivera-Campo.

Chapter 2

Calculating Areas and Volumes

2.1 Introduction

From a historical point of view, it is commonly accepted that early developments in Euclidean geometry were based on the necessity of calculating land areas for tax purposes. Currently we still need to calculate the areas and the volumes of more complicated objects; for example, cartographic maps or volumes of plastic objects for mass production.

The main objective of this chapter is to study the problems related to measuring the areas and the volumes of polygons and polyhedra in both the plane and the three dimensional space.

Let P_n be a simple polygon with n vertices. The traditional method to calculate its area consists in dividing P_n into triangles and then finding the area of each triangle and adding them together:

Algorithm 2.1.1 *Area using triangulation*

1. *Triangulate* P_n
2. *Measure and sum the areas of the* $n-2$ *triangles of the triangulation of* P_n.

It is not difficult to see that the complexity of this algorithm is linear. The first step of the process is to triangulate P_n, which can be done in linear time using an algorithm by Chazelle [Cha91] to triangulate polygons. However, Chazelle's algorithm has a drawback; it is extremely difficult to implement. From a practical point of view, we could use a traditional $O(n \log n)$ time triangulation algorithm that would result in a practical $O(n \log n)$ time algorithm.

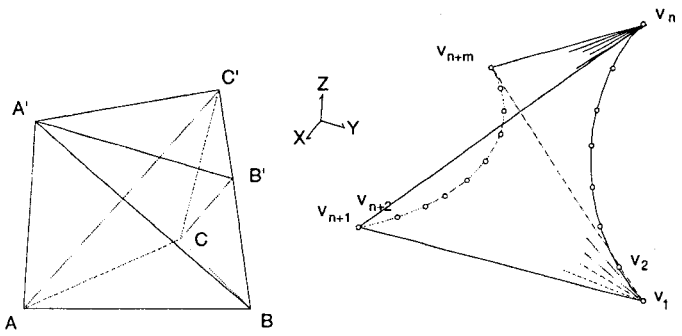

Figure 2.1: (a) A Lennes polyhedron is not decomposable into tetrahedra. (b) A polyhedron decomposable into a quadratic number of tetrahedra.

Another inconvenience of Algorithm 2.1.1, perhaps the most important one, resides in the fact that it is not generalizable to three dimensions. If we consider decompositions of polyhedra into tetrahedra, which is the natural

generalization of triangulations, it is not true that any polyhedron Q_n in $\boldsymbol{R}^3$ can be decomposed into tetrahedra such that the vertices of each tetrahedron are vertices of Q_n. For example, in Figure 2.1a, a Lennes polyhedron is constructed as follows: $A'B'C'$ is a copy of the equilateral triangle ABC, with sides of length one, rotated 30° in its plane. It is parallel to the plane of ABC and separated by a distance of one unit. Here, the segment that joins two non-adjacent vertices lies outside the polyhedron, therefore a tetrahedron using it is not contained in the polyhedron (see [Eve72]).

Yet another problem comes from the fact that there exist some polyhedra such that any decomposition into tetrahedra (possibly with vertices in the interior of Q_n) has a quadratic number of elements. See, for example, Figure 2.1b where $\{v_1, v_2, \ldots, v_n\}$ are n vertices contained in the plane YZ and $\{v_{n+1}, v_{n+2}, \ldots, v_{n+m}\}$ are m vertices contained in the plane XY. Here, the only tetrahedrization possible consists of tetrahedra of the form $\{v_{n+i}, v_{n+i+1}, v_j, v_{j+1}\}$, for $1 \leq i < m, 1 \leq j < n$, which are $O(n \cdot m)$. This problem shows that this kind of decomposition to calculate volumes would imply an algorithm of time complexity higher than linear.

Such difficulties obligate us to re-examine our problem. A different and perhaps easier approach will be shown in the next section. Its generalization to higher dimensions will be shown in Section 2.3. We will observe that the complexity for at least two and three dimensions is linear.

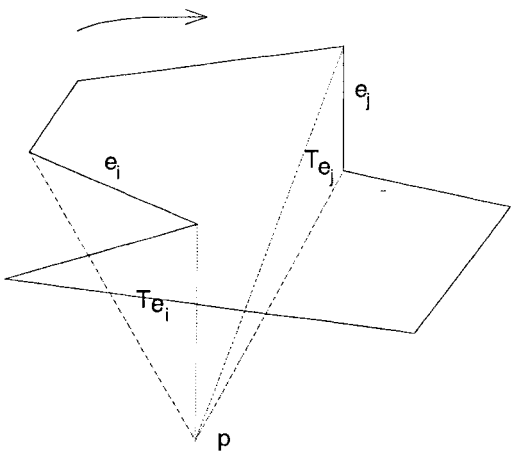

Figure 2.2: A_{e_i} is negative and A_{e_j} is positive.

2.2 A generalizable method

Let p be any point in the plane and e_i an edge of P_n. Every edge e_i of P_n defines a line L_{e_i} dividing the plane into two half-planes, $P^+_{e_i}$ and $P^-_{e_i}$, in such a way that $P^+_{e_i}$ is the half-plane to the right of e_i when the boundary of P_n is traversed clockwise. Let T_{e_i} be the triangle defined by e_i and p, and let a_{e_i} be its area. Define A_{e_i} as follows: if $p \in P^+_{e_i}$ then $A_{e_i} = a_{e_i}$, otherwise $A_{e_i} = -a_{e_i}$. If $\mathcal{A}(P_n)$ is the area of P_n, then the following equality can be obtained from *Jordan's closed curve theorem*:

$$\mathcal{A}(P_n) = \sum_{i=1}^{n} A_{e_i} \tag{2.1}$$

Indeed, if R is one of the regions defined by the edges of the triangles T_{e_i} and $\mathcal{A}(R)$ its area, we have to show the following: If $R \subset P_n$ then $\mathcal{A}(R)$ is added only once to the summation in Equation 2.1, otherwise if $R \not\subset P_n$, its area $\mathcal{A}(R)$ will be canceled by another term in the summation. That is, if R

is contained in a triangle T_{e_j} with positive area A_{e_j}, we will show that there exists another triangle $T_{e_{j'}}$ with negative area $A_{e_{j'}}$ such that $R \subset T_{e_{j'}}$. In this way we will show that the summation in Equation 2.1 gives the area of the polygon P_n.

Let q be a point inside R and let $\overrightarrow{r}$ be the (infinite) portion of the ray $\overrightarrow{pq}$ to the right of q, when p is to the left of q. Then $\overrightarrow{r}$ may intersect some edges of P_n. If it does, it will intersect exactly those edges corresponding to triangles T_{e_i} containing R.

Observation 2.2.1 *Let e_i be the first edge crossed by $\overrightarrow{r}$. If there exists a point $x \in P_n$ between q and e_i, then both p and q are contained in $P_{e_i}^+$ (in particular, q is interior to P_n), otherwise both p and q are contained in $P_{e_i}^-$.*

This observation comes from the fact that the interior of a polygon lies to the right of each edge when the edges of the polygon are oriented clockwise. See, for example [Mun75].

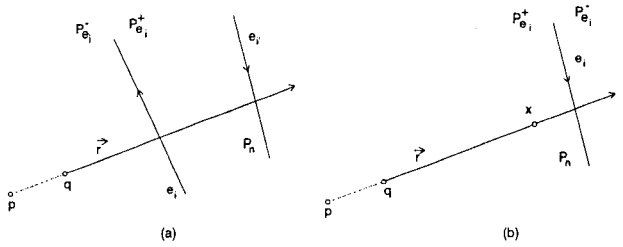

Figure 2.3: The position of p and q with respect to e_i.

Suppose first that $R \not\subset P_n$. Given that the boundary of a simple polygon is a bounded and closed curve (also known as a *Jordan curve*, see [Mun75]), whenever the ray $\overrightarrow{r}$ crosses an edge e_j to enter P_n, we have a triangle T_{e_j}

containing R, with negative area because of Observation 2.2.1. That is, we are adding $-\mathcal{A}(R)$ to the summation. But the same Jordan's closed curve theorem tells us that P_n has an exterior region and since the polygon is bounded, the ray $\overrightarrow{r}$ has to exit to the exterior of P_n crossing another edge $e_{j'}$. Again, by Observation 2.2.1, $p \in P^+_{e_{j'}}$. Therefore the area of triangle $T_{e_{j'}}$ is positive and this implies that we have to add $+\mathcal{A}(R)$ to the summation, in this way canceling the previous area of R. That is, if $R \not\subset P_n$, the area of the region R does not contribute to the summation in Equation 2.1. The same procedure can be applied each time the ray $\overrightarrow{r}$ crosses the boundary of P_n; the area of R will be canceled when it enters and leaves P_n.

We have an similar situation if $R \subset P_n$, but in this case we have to add $+\mathcal{A}(R)$ to the summation in Equation 2.1 once, because for the first edge e_i crossed by ray $\overrightarrow{r}$, our Observation 2.2.1 tells us that the triangle T_{e_i} has positive area. Now we move the point q along $\overrightarrow{r}$ until it lies outside P_n and we have the previous situation for this new $\overrightarrow{r}$. That is, a further addition of $-\mathcal{A}(R)$ to the summation in Equation 2.1, if any, will be canceled by an addition of $\mathcal{A}(R)$.

Since the area of P_n is the sum of the areas of regions R contained in P_n we have:

Algorithm 2.2.1 *Area of P_n*

1. *Choose a point p in the plane*
2. *For each edge e_i of P_n, do:*

   Calculate the area A_{e_i} of triangle T_{e_i}
3. *Sum over all A_{e_i}*

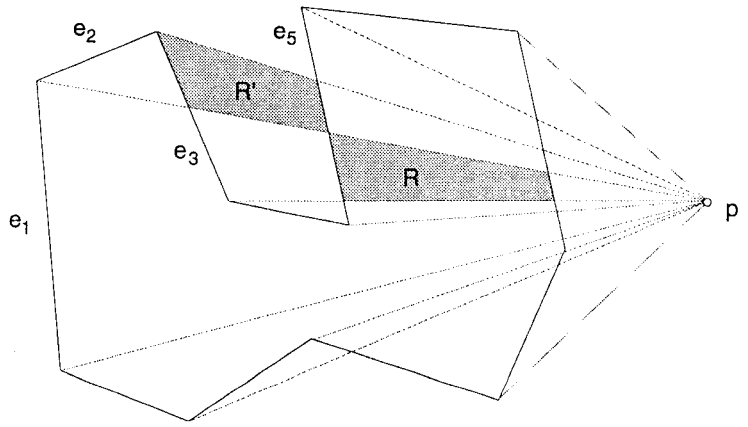

Figure 2.4: Two cases: R (inside P_n) is contained in T_{e_1}, T_{e_3} and T_{e_5} and R' (outside P_n) is contained in T_{e_2} and T_{e_3}.

This algorithm clearly runs in linear time. A point can be chosen in constant time. Furthermore, since Steps 2 and 3 are two loops iterating as many times as the number of edges of the polygon, performing a constant number of operations in each iteration, both of them run in linear time, therefore:

Theorem 2.2.1 *The area $\mathcal{A}(P_n)$ of the polygon P_n can be calculated in linear time.*

Note: As mentioned in Chapter 1, the proof of the correctness of this algorithm, using for example Green's theorem, is well known. One can refer to that chapter to see some widely used "formulas" to calculate the area of a simple polygon.

2.3 Three dimensions

As mentioned previously, this method of measuring areas can be generalized to three dimensions. That is, given the face F_i of an n-vertex polyhedron $Q_n \in \boldsymbol{R}^3$ and a point p also in $\boldsymbol{R}^3$, we proceed to calculate the area $\mathcal{A}(F_i)$, of the face F_i, with the algorithm just shown. We thus obtain the volume v_{F_i} of the pyramid Π_{F_i} (see Figure 2.5) determined by F_i and the point p, using the formula:

$$v_{F_i} = \frac{\mathcal{A}(F_i) \cdot h_{F_i}}{3} \tag{2.2}$$

where h_{F_i} is the height of p over F_i.

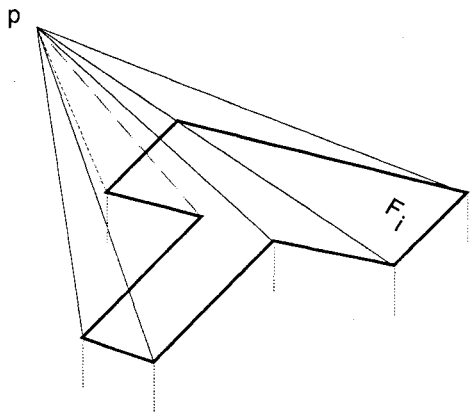

Figure 2.5: Pyramid Π_{F_i}.

For each face F_i we can determine the orientation of its two sides, either outward or inward, by labeling the half spaces determined by the plane P_{F_i} containing F_i, with the following procedure:

Trace the ray $\overrightarrow{r}$ from a point p' outside Q_n towards the polyhedron (see

Figure 2.6). Let F_j be the first face of Q_n intersected by $\overrightarrow{r}$ at the point p''.

Obtain the normal vectors n_j^L and n_j^R at both sides of F_j, both departing from p''. Assign the label $S_{F_j}^-$ to the half space determined by P_{F_j} at the side of the normal vector determining the smallest angle with vector $p''p'$. Suppose, for example that n_j^L is such a vector determining a smaller angle with $p''p'$ than n_j^R. It can be seen that the side of F_j obtained in this way always faces outwards. Assign the label $S_{F_j}^+$ to the complementary half space.

In order to label all the other half spaces determined by faces of Q_n, let F_k be a unlabeled face which is adjacent to a labeled face F_j. Suppose that the normal vector n_j^L points outwards from F_j and obtain the normal vectors n_k^L and n_k^R of F_k. If the smaller angle α of P_{F_j} and P_{F_k} is smaller than 90° then let $m = n_j^R$, otherwise $m = n_j^L$. Now give the label $S_{F_k}^-$ to the half space on the side of the normal vector, n_k^L or n_k^R, making the smallest angle with m. It can be proved that this side of F_k always faces outwards. Again, give the label $S_{F_k}^+$ to the complementary half space.

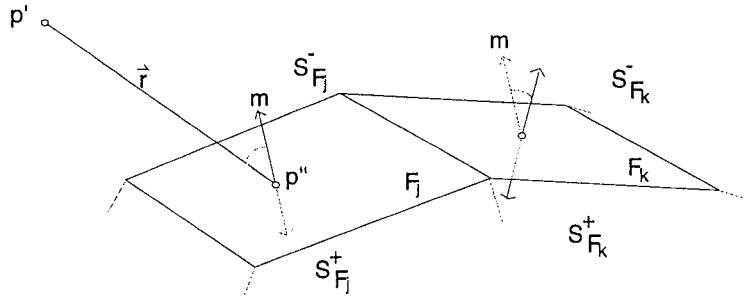

Figure 2.6: $S_{F_k}^-$ and $S_{F_k}^+$.

Now define $V_{F_i} = v_{F_i}$ if $p \in S_{F_i}^+$ and $V_{F_i} = -v_{F_i}$ otherwise. We will observe that the volume $\mathcal{V}(Q_n)$ of Q_n can be calculated by:

$$\mathcal{V}(Q_n) = \sum_{i=1}^{f_n} V_{F_i} \tag{2.3}$$

where f_n is the number of faces of Q_n.

The proof that this summation does calculate the volume of Q_n is similar to the previous analysis for the planar case. That is, we observe that the volume of a region R determined by faces of the pyramids P_{F_i} and faces of Q_n is counted either only once or not at all, depending on whether R is contained in Q_n or not, because its volume can be canceled. The details are the same as above because the boundary of any simple polyhedron is a *Jordan variety*, see [Mun75]. Therefore our algorithm is as follows:

Algorithm 2.3.1 *Volume of the polyhedron Q_n*

1. *Label the sides of the face F_i of Q_n as $S^-_{F_i}$ and $S^+_{F_i}$ as explained.*

2. *For each face F_i of the polyhedron Q_n, do:*

   Calculate the volume V_{F_i} of the pyramid Π_{F_i} using Equation 2.2.

3. *Sum over all V_{F_i}*

Now we will show that the time complexity of this algorithm is linear in n. To prove this, observe that if a face F_i has $v(F_i)$ vertices, then its area can be calculated in time $O(v(F_i))$. Calculating the volume v_{F_i} does not modify this complexity.

Going through the faces of Q_n searching for the first face to intersect with $\overrightarrow{r}$ takes only linear time. This is done only for the first face. To decide which side of each face the point p belongs to, takes constant time.

Finally, the summation of the volumes in Equation 2.3 also takes linear time, because the number of faces is linear. This is so because the degree of each vertex is at least three. Using Euler's formula (see for example [PS85]), we then obtain the result.

Theorem 2.3.1 *Measuring the volume $\mathcal{V}(Q_n)$ of any simple polyhedron Q_n in $\boldsymbol{R}^3$ can be done in linear time.*

Chapter 3

Areas determined by linear cuts

A farmer has an extensive land property which she wants to divide into two parts. One part might be used for livestock and the other for crops, for example. The choice for the dividing line could be made on a number of different criteria such as the kind of terrain, soil quality, water sources, etc. For each possible choice, she needs to know the exact area of each of the two parts.

In this chapter, we study an abstraction of the problem: Suppose that we have a convex polygon in the plane which we cut into two pieces along a line segment. How fast can we calculate the area of the pieces?

Theorem 2.2.1 tells us that this problem can be solved in linear time. However we will obtain a significant improvement by allowing some preprocessing, which will be shown below.

The importance of improving the time complexity is clear when the number of cuts increases. Therefore, we want to find a method to preprocess the polygon (storing partial areas) such that any further calculation of the areas

can be done in less than linear time.

3.1 Cutting convex polygons

Let P_n be a convex polygon with vertices $\{v_1, \ldots, v_n\}$ given in clockwise order such that v_1 is the vertex with the lowest y-coordinate. Preprocess P_n as follows:

Algorithm 3.1.1 *Preprocessing convex polygons*

1. *Let T_{v_1} be the triangulation of P_n given by joining v_1 with all other vertices of P_n. Let T_i be the triangle with vertices $\{v_1, v_{i+1}, v_{i+2}\}$ and let a_i be its area, for $i = 1, \ldots, n-2$.*

2. *Let $A_1 = 0, A_2 = 0, A_i = a_1 + \cdots + a_{i-2}$, for $i = 3, \ldots, n$*

Now suppose that we cut P_n into two pieces along a line segment joining vertices v_i and v_j, $i < j$. Let $S_{i,j}$ be the piece of P_n containing v_1. Then the area $\mathcal{A}(i,j)$ of $S_{i,j}$ is given by the following formula:

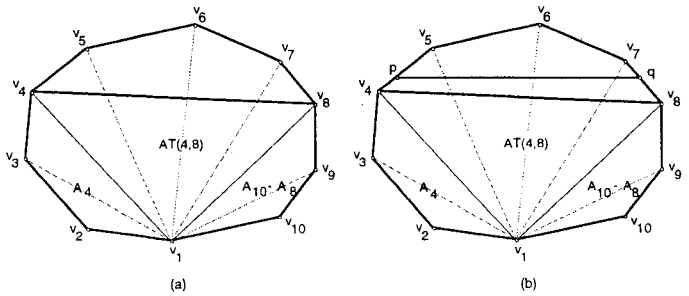

Figure 3.1: Calculating the area (a) below $\overline{v_4 v_8}$ and (b) below $\overline{pq}$.

$$\mathcal{A}(i,j) = A_i + (A_n - A_j) + AT(i,j) \tag{3.1}$$

where $AT(i,j)$ is the area of the triangle with vertices v_1, v_i, v_j. Therefore, we have:

Theorem 3.1.1 *Given $A_1, \ldots, A_n$ as in Algorithm 3.1.1, we can calculate $\mathcal{A}(i,j)$ in constant time.*

Now suppose that instead of cutting P_n along a line segment joining two vertices, we want to cut it anywhere using a straight-line cut.

To solve this problem in logarithmic time, recall that the intersection points p and q of a line with the boundary of a convex polygon can be found in logarithmic time (see [PS85]). Suppose p and q lie on the edges e_i and e_{j-1} of P_n, joining vertices v_i to v_{i+1} and v_{j-1} to v_j respectively. Then the area of the piece of P_n containing v_1 is the area $\mathcal{A}(i,j)$ plus the area of the convex quadrilateral of vertices $\{v_i, v_j, q, p\}$. By the previous theorem, $\mathcal{A}(i,j)$ can be calculated in constant time and, since the area of a convex quadrilateral can also be calculated in constant time, we have:

Theorem 3.1.2 *Given $A_1, \ldots, A_n$ as in Algorithm 3.1.1, the areas of the pieces of a convex polygon split by a line can be calculated in logarithmic time.*

We will find the area of a simple polygon cut by an interior segment in the following chapter. Here we will continue with an interesting application of the result just shown.

3.2 Bisecting convex polygons

In an animated cartoon, suppose we have a character balancing on a log floating on the river. Each time the unlucky guy takes a step, the log rolls in place.

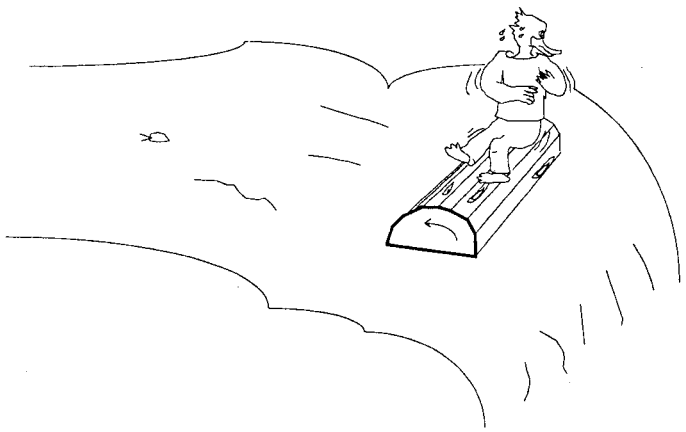

Figure 3.2: Bisecting a convex polygon may save a life.

The log is submerged in the water up to a certain level which we want to determine, in order to draw its visible portion for each angle of the rolling.

From basic physics we know that the ratio ρ between the volume submerged and the total volume of the log is kept constant, since in our simplistic model, it depends only on the sum of the *densities* of both objects, the character and the log.

If the log is modeled by a prism having the convex polygon P_n as its base, then we need to calculate the level of the water dividing a portion ρ of the area of P_n for each rotation angle α.

Since the character is in a hurry because he is trying to escape by rolling the log away from a waterfall, we need to calculate and draw such levels very

quickly.

Let us consider the case when $\rho = 1/2$. We have the following equivalent problem:

Problem 3.2.1 *Given a convex polygon P_n and a line L, find a line L' parallel to L bisecting P_n, i.e. splitting P_n into two pieces of equal area.*

We start by supposing that the line L is parallel to the x-axis and then we modify this assumption to use any line L afterwards.

We will suppose that the areas $A_1, \ldots, A_n$ have been calculated in pre-processing time with Algorithm 3.1.1. Suppose that the vertices of P_n are labeled clockwise as $\{v_1, \ldots, v_k, \ldots, v_n\}$ such that v_1 and v_k are the vertices of P_n with the smallest and largest y-coordinate respectively.

We begin by showing how to perform a binary search to find the edges of P_n intersected by the horizontal line splitting P_n into two pieces of equal area.

The main idea of our algorithm is the following: In each iteration we consider two chains of edges of P_n, namely C_R and C_L on the right and left sides of P_n respectively, such that initially, their union is exactly all the edges of P_n. Our objective is to eliminate half the edges of one of the chains in each iteration. Let i and j be subindexes of the first and last vertices of C_R and let l and m be subindexes of the first and last vertices of C_L. Initially $i = 1$, $j = k$, $l = k$ and $m = n + 1$. Since P_n has n vertices, we let $v_{n+1} = v_1$. Our binary search will stop at the moment when either C_R or C_L has one edge. We proceed as follows:

Algorithm 3.2.1 *Reducing the length of the chains*

While C_R and C_L have more than one edge, do:

1. *Let* $r = \lfloor \frac{l+m}{2} \rfloor, s = \lfloor \frac{i+j}{2} \rfloor$
2. *If the slope of* $\overline{v_r v_s}$ *is positive then*

   If $\mathcal{A}(r,s) \leq \mathcal{A}(P_n)/2$ *then* $m = r$, *else* $j = s$
3. *If the slope of* $\overline{p_r p_s}$ *is negative then*

   If $\mathcal{A}(r,s) \leq \mathcal{A}(P_n)/2$ *then* $i = s$, *else* $l = r$

Our search process has to be modified slightly when one of our chains contains only one edge. For this case, we need to reduce the length of the other chain, keeping the condition:

$$BoundingBox(l,m,i,j) : \mathcal{A}(m,i) < \mathcal{A}(P_n)/2 \leq \mathcal{A}(l,j) \tag{3.2}$$

with $i < j$ and $l < m$. This means that $\overline{v_m v_i}$ is always above L' and $\overline{v_l v_j}$ is always below.

We proceed as follows (here a chain C has $|C|$ edges):

Algorithm 3.2.2 *Finding the bounding box*

1. *While* $|C_L| > 1$ *and* $BoundingBox(l,m,i,j)$

   (a) $r = \lfloor \frac{l+m}{2} \rfloor$

   (b) *If the slope of* $\overline{v_r v_i}$ *is negative then* $l = r$, *else* $m = r$

   (c) *If the slope of* $\overline{v_r v_j}$ *is positive then* $m = r$, *else* $l = r$

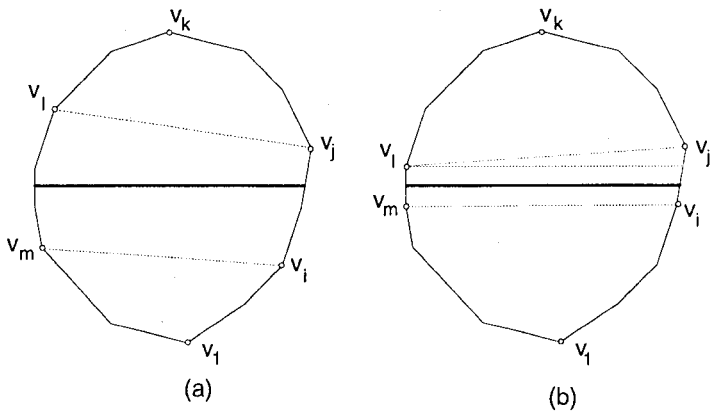

Figure 3.3: The bounding box, initially with several edges (a), becomes a quadrilateral with two horizontal edges (b).

---

2. *While* $|C_R| > 1$ *and* $BoundingBox(l, m, i, j)$

   (a) $s = \lfloor \frac{i+j}{2} \rfloor$

   (b) *If the slope of* $\overline{v_m v_s}$ *is negative then* $i = s$, *else* $j = s$

   (c) *If the slope of* $\overline{v_l v_s}$ *is positive then* $j = s$, *else* $i = s$

With this procedure, we ensure that a quadrilateral Q with vertices $\{v_i, v_j, v_l, v_m\}$ becomes the final bounding box. In this way, the line L' cuts edges $\overline{v_i v_j}$ and $\overline{v_l v_m}$ of Q and we have:

Lemma 3.2.1 *We can find the edges e and e' of* P_n *intersected by the horizontal line bisecting* P_n *with a logarithmic number of steps.*

The number of steps is logarithmic because our procedure in both algorithms is to reduce the number of edges in the bounding box by half at each

iteration, and since P_n is convex, the line L' will intersect exactly two of its edges $\overline{v_i v_j}$ and $\overline{v_l v_m}$ as shown.

Without loss of generality, we can assume that $\overline{v_l v_j}$ and $\overline{v_m v_i}$ are horizontal. Otherwise, we can calculate the largest quadrilateral with two horizontal edges contained in Q in constant time. We can make this assumption because our line L' is horizontal, reducing our problem to:

Problem 3.2.2 (The bounding box) *Let Q be a convex quadrilateral and assume a and b are two opposite edges parallel to the x-axis, and let two regions A and B have areas $\mathcal{A}(A)$ and $\mathcal{A}(B)$ respectively. Find a horizontal line L dividing Q into two regions A' and B' with areas $\mathcal{A}(A')$ and $\mathcal{A}(B')$ respectively, such that $\mathcal{A}(A) + \mathcal{A}(A') = \mathcal{A}(B) + \mathcal{A}(B')$.*

This problem can be solved in constant time. Therefore, we have:

Theorem 3.2.1 *To find the horizontal line bisecting a convex polygon P_n takes $O(\log n)$ time, and a linear time preprocessing.*

To solve our problem in the case when L is not parallel to the x-axis, we modify our procedure as follows:

Let v_a and v_b be the first and last vertices intersected by a line sweeping P_n from bottom to top, respectively. The rest of the process is the same as above, except that now v_a replaces v_1 and v_b replaces v_k, and instead of comparing the slope of the line segment joining v_r to v_s against the horizontal, we compare it against the slope of the line L. The vertices v_a and v_b can found in logarithmic time by performing a binary search among the vertices of P_n. Therefore, we have (assume preprocessing using Algorithm 3.1.1):

Algorithm 3.2.3 *Finding the line L' parallel to a line L which splits the area of the convex polygon P_n in half.*

1. *Find vertices v_a and v_b and construct initial chains C_L and C_R.*
2. *Use Algorithms 3.2.1 and 3.2.2, substituting "slope of X" by "slope of $X-$ slope of L", to construct a bounding box Q with two edges parallel to L.*
3. *Solve Problem 3.2.2 for Q.*

and we have:

Theorem 3.2.2 *Using linear preprocessing we can find a line parallel to any given line L bisecting a convex polygon P_n in logarithmic time.*

Notice that the same procedure can be applied to find a line cutting the fraction ρ of the area of the convex polygon, for $\rho < 1$. This is done with few changes:

1. Replace all occurrences of the factor $\frac{1}{2}$ by ρ in Algorithms 3.2.1 and 3.2.2
2. The condition on page 26 (Inequality 3.2) is now:

$$BoundingBox(l, m, i, j) : \mathcal{A}(m, i) < \rho \cdot \mathcal{A}(P_n) \leq \mathcal{A}(l, j)$$

Chapter 4

Cutting non-convex polygons

4.1 Preliminaries

In this chapter we will find an algorithm such that with $O(n \log n)$ preprocessing time and space, we can solve the following problem in logarithmic time (see Figure 4.1a):

Problem 4.1.1 (Cutting simple polygons) *Suppose we cut the polygon P_n into two pieces along a line segment joining two boundary points. Calculate the areas of the regions determined by the line segment.*

In order to solve our problem we reduce it in logarithmic time to the following version (see Figure 4.1b):

Problem 4.1.2 *Let e be a distinguished edge of a polygon P_n called* the base of P_n. *We want to calculate the area of a piece of P_n cut by a line segment with one of its endpoints in e.*

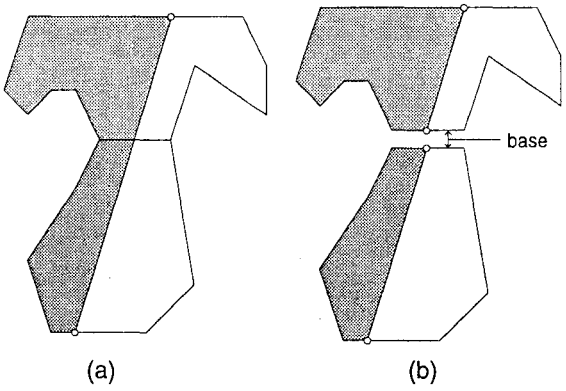

Figure 4.1: (a) Original problem (b) Reduced version.

We first show how to perform the reduction of Problem 4.1.1 into Problem 4.1.2 in logarithmic time and then we proceed to show the solution for Problem 4.1.2 in constant time after a linear preprocessing.

4.2 The first reduction

Let P_n be a polygon with n vertices $\{v_1, \ldots, v_n\}$ given in clockwise order and let $bd(P_n)$ be its *boundary.*

Let H be any triangulation of P_n and $e_1, \ldots, e_{n-3}$ be the diagonals of P_n used in H. We associate to each e_i in $\{e_1, \ldots, e_{n-3}\}$ the subpolygons $P^L_{e_i}$ and $P^R_{e_i}$ of P_n as follows:

Algorithm 4.2.1 *Partition(P_n, H)*

While P_n has more than 3 edges do:

1. *Find an edge e_i in $\{e_1, \ldots, e_{n-3}\}$ cutting P_n into two subpolygons $P^L_{e_i}$ and $P^R_{e_i}$, having p_L and p_R vertices respectively with $p_L \leq p_R$, such that*

$\lfloor \frac{n}{3} \rfloor \leq p_L + 1$ *and* $p_R \leq \lfloor \frac{2n}{3} \rfloor + 1$.

2. *Edge* e_i *will be* the base *of* $P^L_{e_i}$ *and* $P^R_{e_i}$.

3. *Partition*($P^L_{e_i}$, H'), *Partition*($P^R_{e_i}$, H'')

where H' and H'' are the triangulations of $P^L_{e_i}$ and $P^R_{e_i}$ respectively induced by H. The number of iterations that this algorithm makes is logarithmic. The existence of e_i is a direct consequence of the known $\frac{1}{3} - \frac{2}{3}$ theorem [Cha82], which essentially states that any polygon has a diagonal dividing it into two pieces of similar size.

Algorithm 4.2.1 also defines a binary tree T with logarithmic depth in which we will store the diagonals $\{e_1, \ldots, e_{n-3}\}$ in H and the area $\mathcal{A}(e_i)$ of P_n at one side of each diagonal, using the following recursive step:

In the root of T store the edge e_i selected by Partition(P_n, H). Then the left son of e_i, if it exists, will be the diagonal e^L_i of $P^L_{e_i}$ selected by Partition($P^L_{e_i}$, H') and the right son, also if it exists, will be the diagonal e^R_i of $P^R_{e_i}$ selected by Partition($P^R_{e_i}$, H''); see Figure 4.2.

The area $\mathcal{A}(P^L_{e_i})$ to the left of e_i will be calculated recursively as:

$$\mathcal{A}(P^L_{e_i}) = \mathcal{A}(P^L_{e^L_i}) + \mathcal{A}(P^R_{e^L_i})$$

if e^L_i exists, and $\mathcal{A}(P^L_{e_i})$ ="area of triangle to the left of e^L_i", if it does not.

Now we will show how to use T to reduce Problem 4.1.1 into Problem 4.1.2 in logarithmic time. Suppose that we cut P_n along a line segment c joining the points p and q in $bd(P_n)$, and we want to calculate the areas of the resulting polygons.

We proceed to do a search in T to find the first diagonal of H intersecting c, starting from the root of T. First we check if c intersects the diagonal e_i

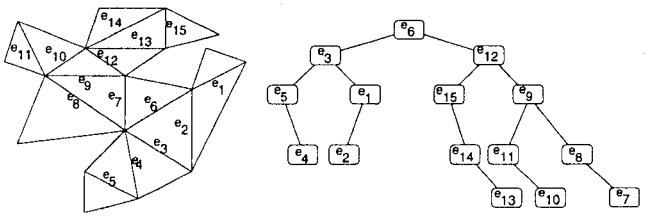

Figure 4.2: Constructing the binary tree T.

stored at the root of T; this can be done in constant time by checking if these two segments intersect each other. If that is the case, our problem is reduced to calculating the areas of the polygons generated by cutting $P_{e_i}^L$ and $P_{e_i}^R$ along c. This procedure requires us to solve two instances of Problem 4.1.2; one for the polygon $A = P_{e_i}^L$ and another for the polygon $B = P_{e_i}^R$, both with base e_i and cut c.

In the case that c does not intersect e_i, we must first determine whether c is contained in $P_{e_i}^L$ or in $P_{e_i}^R$. This can be done in constant time as follows: Assuming that P_n has its vertices sorted in clockwise order, let $e_i = \overline{v_j v_k}$ be the diagonal to compare, and let $\overline{v_l v_{l+1}}$ and $\overline{v_{m-1} v_m}$ be the edges of P_n intersected by c. Then, if we denote $[a] = (a \bmod n)$, we have that since e_i and c do not intersect each other, $[m-l] \leq [k-j]$. If $[l-j] < [k-j]$ then c is in $P_{e_i}^L$, else it is in $P_{e_i}^R$.

If c is in $P_{e_i}^L$, we will search for a diagonal of $P_{e_i}^L$ intersecting c in the subtree determined by the left son of e_i. Otherwise, if c is contained in $P_{e_i}^R$, we will search in the subtree determined by the right son of e_i in T.

From here we will proceed recursively, traversing T searching for the first diagonal e_k intersecting c. It could happen that c does not intersect any

diagonal in H. This is the case when c cuts a triangle Δ through a vertex or it cuts a leaf triangle Δ. For both cases we only need to know the area of Δ cut by c and the area stored at the corresponding diagonal of P_n determining an edge of Δ.

Suppose that we have found the first diagonal e_k intersected by c using the process shown above. We will obtain the subpolygons $S^L_{e_k}$ of $P^L_{e_k}$ and $S^R_{e_k}$ of $P^R_{e_k}$ determined by the first diagonal l_k to the left and the first diagonal r_k to the right of e_k respectively among all the parent diagonals of e_k in T. Neither l_k nor r_k are intersected by c, therefore we have essentially two cases for l_k and r_k: they are contained in different portions of P_n cut by c or they belong to the same portion (see Figure 4.3).

In either case, we have two instances of Problem 4.1.2, obtaining the areas $\mathcal{A}(A)$ and $\mathcal{A}(B)$ of polygons $A = S^L_{e_k}$ and $B = S^R_{e_k}$, both with base e_k and cut c. The area $\mathcal{A}(c)$ to the left of c is calculated as (see Figure 4.3)

$$\mathcal{A}(c) = \mathcal{A}(A) + \mathcal{A}(B) + \mathcal{A}(l_k)$$

(supposing l_k is to the left of c) for the first case, or

$$\mathcal{A}(c) = \mathcal{A}(A) + \mathcal{A}(B) + \mathcal{A}(l_k) + \mathcal{A}(r_k)$$

(supposing both diagonals are to the left of c) for the second case.

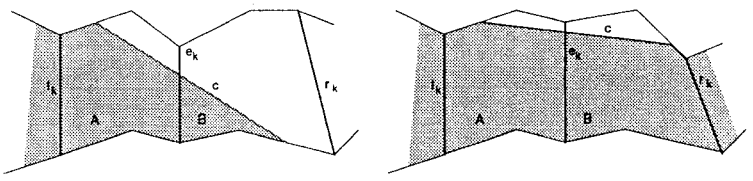

Figure 4.3: Applying the reduction to solve Problem 4.1.1.

Theorem 4.2.1 *Problem 4.1.1 can be reduced to Problem 4.1.2 in $O(\log n)$ time and space.*

For each diagonal e_i of H, we will show how to preprocess the two sub-polygons $S^L_{e_i}$ and $S^R_{e_i}$ of P_n in constant time using linear space and time preprocessing. If we add together the global space and time complexity of the calculation of the areas to the left of each diagonal in H, and of the procedures to triangulate and to construct the tree T, we have an overall time and space complexity for the preprocessing in our algorithm of:

$$\sum_{e_i \in H} O(\left|P^L_{e_i}\right|) + O(\left|P^R_{e_i}\right|) = O(n \log n) \tag{4.1}$$

In the following sections we will show the guidelines to solve Problem 4.1.2 in constant time using linear preprocessing.

An edge e' of P_n is called *weakly visible* from e if there exists a segment ℓ interior to P_n joining e' to e. In such a case, the segment ℓ will be called a *link* of e' to e (some authors refer to e as the *visibility segment*). We allow ℓ to touch other points on $bd(P_n)$ without intersecting the exterior of P_n.

Let $c = \overline{pq}$ be a query cut, that is, c is an segment interior to P_n with p on e and q on e', where e' is weakly visible from e. Observe that the edge e' can be obtained in logarithmic time if q is given.

Observation 4.2.1 *Note that our query cut c only intersects edges weakly visible from the base, due to our assumption that it cuts the polygon into only two pieces.*

We will divide our work as follows:

In Section 4.3 we find the weakly visible edges from e.

In Section 4.4 we determine the links for each weakly visible edge.

In Section 4.5 we find the area to the left of each link.

In Section 4.6 given the query cut c, we find its corresponding link $\ell(e')$, then we calculate the area to the left of c from the areas obtained in Section 4.5.

4.3 Edges weakly visible from the base

We will suppose that the base e is horizontal and that the interior of P_n is above e. We need to find all the edges of P_n weakly visible from the base e. Our solution to this problem, although simple, requires a triangulation of P_n.

Note: The procedure described below can also be used to construct the *Weak Visibility Polygon* of P_n from e (see for example [GMP+93]), which is the polygon having all of its edges visible from e (see Figure 4.5).

We will show that all the edges e' of P_n visible from the base[1] $e = \overline{ab}$ can be found in linear time using two *Shortest Path Trees (SPTs)* T_a and T_b with roots at a and b respectively (see Figure 4.4).

The *Shortest Path Tree* of a polygon P_n rooted at a is the union of the (Euclidean) shortest paths from each vertex of P_n to a, using only interior segments (also called *geodesic paths*) to P_n.

[1]We will always assume that all our edges except the base e, are named in clockwise order of the vertices on $bd(P_n)$. The base $e = \overline{ab}$ is the only edge named in counterclockwise order.

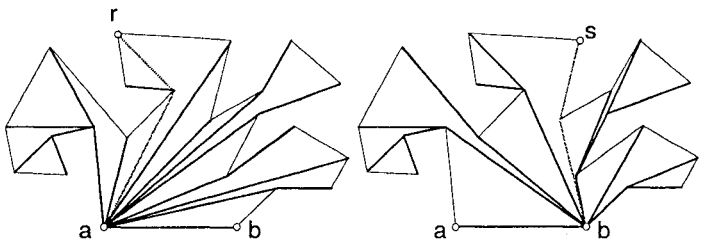

Figure 4.4: Trees T_a and T_b, rooted at the base endpoints. CL_a^r and CR_b^s are in dotted lines.

We will observe that the edges visible from the base are those whose left and right endpoints can be reached by convex and concave chains of edges of the SPTs, respectively.

If the chain of nodes $\{p_1 = a, p_2, \ldots, p_{k_j} = v_j\}$ of T_a from a to v_j (that is, the shortest path from a to v_j) is convex, in other words, p_i is to the left of $\overline{p_{i-2}p_{i-1}}$, for $i = 3, \ldots, k_j$, then we will denote it as $CL_a^{v_j}$ and we call it the *left chain from a to* v_j (see Figure 4.4).

Similarly if the chain of nodes $\{p_1 = b, p_2, \ldots, p_{l_j} = v_j\}$ of T_b from b to v_j (that is, the shortest path from b to v_j) is concave, in other words, p_i is to the right of $\overline{p_{i-2}p_{i-1}}$, for $i = 3, \ldots, l_j$, then we denote it as $CR_b^{v_j}$ and we call it the *right chain from b to* v_j (see Figure 4.4).

Let $V(e) = \{e' = \overline{gd}$ edge of $P_n | CL_a^g \neq \emptyset, CR_b^d \neq \emptyset\}$. In other words, $V(e)$ is the set of edges for which we can find a left chain from a to the endpoint g and a right chain from b to the endpoint d. $V(e)$ is not empty since it contains the edge e' of P_n adjacent to e (if it is above e) or the closest edge crossing the line containing the base (see Figure 4.5).

Observation 4.3.1 *We will consider another set $U(e)$ of vertices of P_n such that a vertex v is in $U(e)$ iff v can be reached by both left and right chains.*

Observe that the edges in $V(e)$ are the edges weakly visible from the base e (see Figure 4.5). Also observe that the vertices in $U(e)$ are visible from the base.

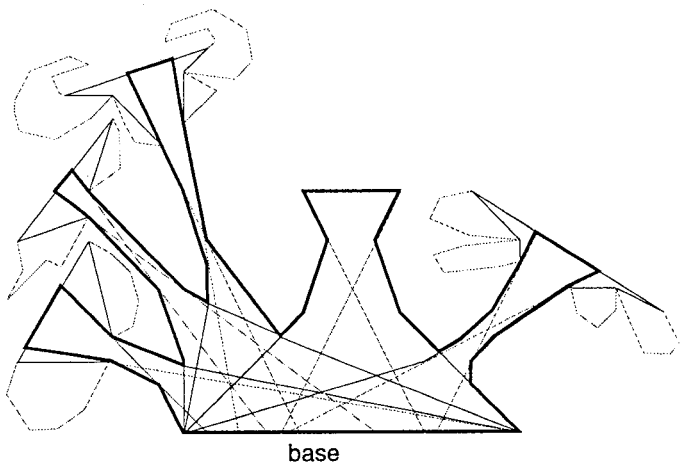

Figure 4.5: Weak Visibility Polygon (thick) of P_n (dotted boundary) from its base, internal segments (dotted) joining visible edges and chains (thin line) defining them.

Before this, we will observe that the left and right chains of a given edge in $V(e)$ do not intersect each other.

Let $e' = \overline{gd}$ be an edge in $V(e)$ (see Figure 4.6a). Suppose CR_b^d contains the vertices v_l and v_m of P_n. First observe that $l < m$. Then if the chain CL_a^g intersects CR_b^d, there exists a vertex v_i of P_n inside the pocket determined by the chain of edges of P_n from v_l to v_m and the edge $\overline{v_m v_l}$. This implies that $\overline{v_m v_l}$ intersects the exterior of P_n, which contradicts the definition of CR_b^d.

Note that if we assume that the vertices of our polygon are in general position, i.e., no three vertices are in a line, we can observe that these chains do not even touch each other.

Furthermore, if we define R as the open region of the plane bounded by CL_a^g, CR_b^d, e and e' (see Figure 4.6b), then R cannot contain any vertex of P_n, otherwise one of the chains would intersect the exterior of P_n which is a contradiction since the chains are interior to P_n. Therefore we have:

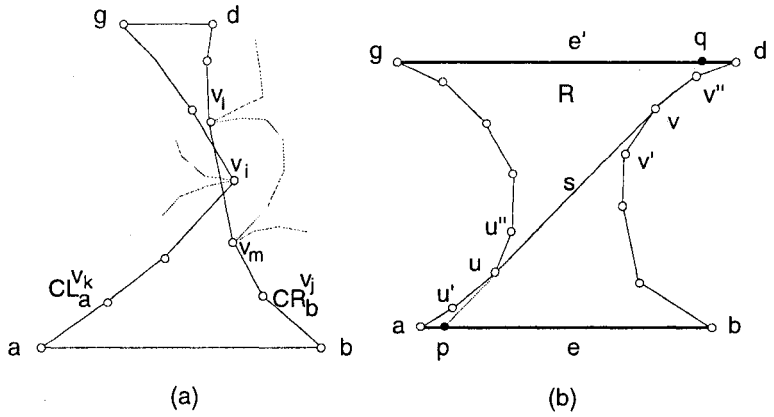

Figure 4.6: (a) The chains CL_a^g and CR_b^d cannot cross each other. (b) The open region R determined by CL_a^g, CR_b^d, e and e'.

Lemma 4.3.1 *If $\overline{gd}$ is an edge of $V(\overline{ab})$ then the left and right chains CL_a^g and CR_b^d from a to g and from b to d of T_a and T_b respectively do not cross each other. Furthermore, the region R between them does not contain vertices of P_n.*

To observe that $V(e)$ is the set of edges weakly visible from e, we will find a segment from each edge e' in $V(e)$ to the base by extending one edge

s of the chain C, where C is the shortest path from a to d contained in P_n.

4.4 Linking edges to the base

Notice that the chain C shares some edges with CL_a^g and some with CR_b^d. Let s be the subchain of C contained in the region R. Notice that s is exactly one edge of C because the vertices of C are vertices of P_n and there are no vertices of P_n in R (see Lemma 4.3.1). Let $s = \overline{uv}$ be the only segment of C in R, where u is a vertex of CL_a^g and v is a vertex of CR_b^d (the segment s is sometimes called an *eave*).

We will show that if we extend the segment s until it hits $bd(P_n)$, it will still be an interior segment of P_n and will join a point p of e with a point q of e', accomplishing the properties of a link (see Page 36).

The *extension* $\chi(s)$ of an edge s of T_a is the segment determined by extending s from both endpoints until it hits the boundary of P_n.

The segment $\ell = \chi(s)$ will not intersect the exterior of P_n because it is a *support segment* (at the points u and v) of both chains. This can be shown as follows:

If u' is the predecessor node of u in the chain CL_a^g and v'' is the successor node of v in the chain CR_b^d (see Figure 4.6b), then u' is to the left of ℓ and v'' is to the right of ℓ, otherwise a chain going from a to d passing through segment $\overline{u'v}$ or segment $\overline{uv''}$ respectively, would be shorter than C. Also if u'' is the successor node of u in the chain CL_a^g and v' is the predecessor node of v in the chain CR_b^d, then u'' is to the left of ℓ and v' is to the right of ℓ, otherwise a part of C cannot be interior to P_n and therefore it cannot be a

branch of the tree T_a.

Thus CL_a^g and CR_b^d are at both sides of ℓ. Reciprocally, a convex and a concave chains can be constructed from the base to each visible edge, therefore we have:

Lemma 4.4.1 *The edge $e' = \overline{gd}$ of P_n is weakly visible from the base $e = \overline{ab}$ if we can reach both endpoints from the base with the chains CL_a^g and CR_b^d. Furthermore, we can find a segment joining e and e', extending a support segment of both chains, which is also an edge in the shortest path from a to d.*

In the previous section, we described the set $V(e)$ of the edges of P_n and this lemma shows a characterization for these edges individually. Now we will show the process to find all the visible edges and their links in linear time.

We first identify all the edges of $V(e)$, that is, all those edges of P_n whose left endpoint can be reached by a left chain from a and whose right endpoint can be reached by a right chain from b, where a and b are the endpoints of the base. At the same time we will identify the vertices reached by both left and right chains, that is, we will create $U(e)$.

For this purpose, we will use a labeling system which assigns to each vertex v of P_n two labels called $\lambda_a(v)$ and $\lambda_b(v)$, where $\lambda_a(v) \in \{\texttt{blank}, \texttt{left}\}$ and $\lambda_b(v) \in \{\texttt{blank}, \texttt{right}\}$. These labels indicate that the vertex has no decision (`blank`), that it belongs to a `left` chain or that it belongs to a `right` chain. In any case, if a vertex v has the label `left` it means that the chain CL_a^v is not empty, similarly if it has the label `right` it means that the chain CR_b^v is not empty.

We will assume that trees T_a and T_b are given. We denote the predecessor node of vertex v in tree T_ξ by $\mathcal{P}_\xi(v)$, and denote $\mathcal{P}_\xi(\mathcal{P}_\xi(v))$ by $\mathcal{P}_\xi^2(v)$.

Algorithm 4.4.1 *Label(P_n)*

Labelling procedure to detect in constant time (in Algorithm 4.4.2) if a vertex is part of a left or right concave/convex chain. Suppose v is a vertex of P_n.

1. $\lambda_a(a)$ =`left`; $\lambda_b(b)$ =`right`;
2. *If* $\mathcal{P}_a(v) = a$, *then* $\lambda_a(v)$ =`left`;

   If $\mathcal{P}_b(v) = b$, *then* $\lambda_b(v)$ =`right`;
3. *If v is to the left of segment* $\overline{\mathcal{P}_a^2(v)\mathcal{P}_a(v)}$ *and* $\lambda_a(\mathcal{P}_a(v))$ =`left`, *then*

   $\lambda_a(v)$ =`left`,

   else $\lambda_a(v)$ =`blank`;
4. *If v is to the right of segment* $\overline{\mathcal{P}_b^2(v)\mathcal{P}_b(v)}$ *and* $\lambda_b(\mathcal{P}_b(v))$ =`right`, *then*

   $\lambda_b(v)$ =`right`,

   else $\lambda_b(v)$ =`blank`;

Essentially, what this algorithm says is that Steps 1 and 2 label the roots of T_a and T_b and their immediate neighbors as `left` and `right`. Step 3 says that a node v can be labeled `left` if its predecessor node is also `left` and it is to the left of the segment formed by its two predecessor nodes. Similarly, Step 4 says that a node v can be labeled `right` if its predecessor node is also `right` and it is to the right of the segment formed by its two predecessor nodes. Otherwise v is labeled `blank`.

Since we can traverse all the nodes of our trees with a breadth-first search, this algorithm takes linear time.

To identify the edges of $V(e)$, we find all the edges having one endpoint `left` and the other `right`. Afterwards we identify the vertices in $U(e)$ in a similar way.

Algorithm 4.4.2 *Find the edges of $P_n = \{v_1, \ldots, v_n\}$ weakly visible from the base e.*

1. *Construct the SPTs, T_a and T_b rooted at the endpoints of the base $e = \overline{ab}$ from a triangulation of P_n.*

   // Obtain the chains for every vertex of P_n:

2. *Do Label(P_n) using Algorithm 4.4.1*

   // Identifying edges in $V(e)$:

3. *For $i = 1$ to n, do:*

   Let $e_i = \overline{g_i d_i}$;

   If $\lambda_a(g_i) =$`left` *and $\lambda_b(d_i) =$*`right` *then*

   Insert e_i into $V(e)$

   // Now find all the vertices in $U(e)$:

   Let v_i be a vertex of P_n.

   If $\lambda_a(v_i) =$`left` *and $\lambda_b(v_i) =$*`right` *then*

   Insert v_i into $U(e)$.

Given a triangulation of P_n, which can be done in linear time, we can construct T_a and T_b also in linear time, therefore Step 1 can be done in linear time. We have seen that Step 2 takes linear time. As Step 3 iterates n times, we have:

Lemma 4.4.2 *If we have the shortest path trees rooted at the base of a polygon, we can find all the edges weakly visible from that base in linear time using Algorithm 4.4.2.*

Now we want to associate each edge $e' = \overline{gd}$ in $V(\overline{ab})$ with its corresponding link $\ell(e')$ to e in linear time, using the fact that the link is determined by an edge of the shortest path from a to d (see Lemma 4.4.1). As our tree T_a is the tree of shortest paths, we will extend every edge s in T_a, until it hits $bd(P_n)$ at edge e' in $V(e)$. We will need the following constructions:

We will say that v is the *distal endpoint* of the segment $s = \overline{uv}$ of the tree T_a if the chain of adjacent vertices $\{a, \ldots, v\}$ in T_a contains u. In this case, u will be called the *proximal endpoint* of s. We will say that $\chi^D(s)$ is the *distal extension* of edge s in T_a if it is the longest segment determined by extending s from its distal endpoint until it hits $bd(P_n)$ without intersecting the exterior of P_n. Similarly, $\chi^P(s)$ will be called the *proximal extension* of s if it is the longest segment determined by extending s from its proximal endpoint until it hits $bd(P_n)$ without intersecting the exterior of P_n.

The new graph $\overline{T}_a$ determined by the distal extensions of all the edges of T_a is a SPT of the polygon P_n with the distal endpoints of the segments $\chi^D(s)$ added (see Figure 4.7).

$\overline{T}_a$ is a tree since T_a is connected and if it were to have any cycles, such

cycles have to be convex polygons, one of whose diagonals would improve a shortest path.

Furthermore, $\overline{T}_a$ determines a partition of P_n into a set of triangles with disjoint interiors. This is because the line segments of $\overline{T}_a$ do not intersect each other and only join vertices of P_n. Also, if w_i and w_j are two adjacent vertices in $bd(P_n)$ determined by the line segments t_i and t_j of $\overline{T}_a$, then t_i and t_j share an endpoint, otherwise we could find a vertex p between t_i and t_j which we could use to construct a better SPT. Therefore all the regions are triangles. Since $\overline{T}_a$ has as many edges as T_a (i.e. $n-1$), the number of triangles is linear.

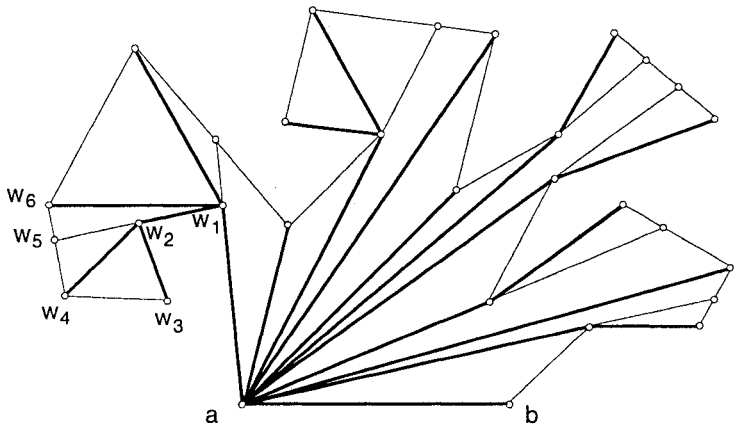

Figure 4.7: The extended tree $\overline{T}_a$.

Lemma 4.4.3 *$\overline{T}_a$ is a SPT of the polygon P_n with distal endpoints added. Furthermore, $\overline{T}_a$ determines a partition of P_n with a linear number of triangles.*

We will now show how to construct the tree $\overline{T}_a$ and the triangulation of P_n determined by its edges, in linear time.

A *funnel* (see Figure 4.7) is a polygon with the following characteristics:

1. A funnel consists of an edge $e' = \overline{gd}$ of P_n called the *lid* of the funnel, and a left and a right chain of edges CL_v^g and CR_v^d from a common vertex v called the *vortex* (e.g. $\{w_1, w_2, w_4, w_6\}$).

2. Funnels whose chains are only two edges of T_a will be called *leaves*, since they correspond to leaves of a tree dual of the triangulation of P_n (e.g. $\{w_2, w_3, w_4\}$).

3. $F_{e'}$ will be called an *empty funnel* if the lid e' is an edge of T_a; that is, if $v = g$ or $v = d$ (e.g. $\{w_2, w_3\}$).

4. The chains CL_v^g and CR_v^d of a funnel $F_{e'}$ are part of the shortest paths from a to g and to d respectively.

Since the SPT T_a is a planar graph, it determines a partition of P_n into subpolygons $F_{e'}$ (see Figure 4.8).

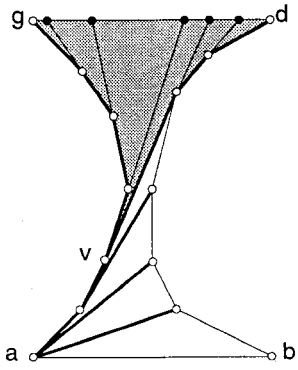

Figure 4.8: A funnel (dark region) determined by T_a (thick lines) with its edges extended (interior thin lines).

Let us verify Characteristic 1 for a subpolygon $F_{e'}$ in the partition determined by T_a. Let $e' = \overline{gd}$ be an edge of P_n. If v is a vertex in the boundary of $F_{e'}$, then v is in the chain C of nodes and edges of T_a ending at g or at d. As a branch of T_a, C is the shortest path from v to either g or d. C must be convex or concave because either $\overline{vg}$ or $\overline{vd}$ or both must intersect either the exterior of $F_{e'}$ or $bd(F_{e'})$. Otherwise if, for example, segment $\overline{vg}$ were to intersect the interior of $F_{e'}$, it would be part of a shorter shortest path from v to g, contradicting the definition of $F_{e'}$. Therefore the two chains defining $F_{e'}$ are convex and concave respectively and $F_{e'}$ is a funnel.

Lemma 4.4.4 *The shortest path tree of a polygon from one of its vertices divides it into a linear number of funnels.*

Although there is a one to one correspondence between the edges of P_n and the funnels, we will not be interested in any empty funnel since its area is zero and the links from it are counted at adjacent funnels.

Since the chains are convex and concave respectively, all the distal extensions of the edges of the funnel $F_{e'}$ intersect e' (see Figure 4.8). In particular, this implies that $\overline{T}_a$ can be found in linear time.

Lemma 4.4.5 *The tree $\overline{T}_a$ determined by the distal extensions of the edges of T_a can be found in linear time.*

We have proved that T_a partitions P_n into funnels and $\overline{T}_a$ partitions each funnel, and therefore P_n, into triangles. With the construction of this triangulation, we are able to determine in linear time which link corresponds to what weakly visible segment in $V(e)$, by using a classical *prune and search* method, pruning one triangle at a time.

4.5 Area preprocessing

The subdivision into triangles and funnels gives us a hint on how to determine the area to the left of each link.

A link is determined by both the distal and the proximal extensions of an edge (the eave) of T_a. Therefore for a given edge s of T_a, we will first calculate the area $\mathcal{A}(\chi^D(s))$ to the left of its distal extension using what we know of $\overline{T}_a$. Afterwards we will add the area $\mathcal{A}(\chi^P(s))$ to the left of its proximal extension.

We will preprocess our triangles in the order given by the dual graph T of the triangulation determined by $\overline{T}_a$. We find T in the traditional way, that is, a node per triangle and an arc between two nodes if the corresponding triangles are adjacent. In this way, T is a tree since P_n has no interior vertices. Furthermore, T is a binary tree if we suppose that no three vertices of P_n are collinear; this is because leaves have degree one (T_1 and T_7 in Figure 4.9), and a triangle of a funnel may be adjacent to two triangles at one edge, to one triangle at the second edge and to none at the third edge (for example, T_3 is adjacent to T_2, T_7 and T_4 in Figure 4.9).

Observe that the dual of the triangulation of a funnel is a chain on T (nodes of degree $\leq$ 2). This implies that we can preprocess funnels and adjacent leaves as in Section 3.1, where the dual of the triangulation given for a convex polygon was also a chain.

We say that v is the *distal endpoint of an arc uv of T*, if the chain of vertices of T from its root to v contains u. In this case, we say that u is the *proximal endpoint of arc uv*. A *distal chain of nodes C from node u* is a chain on T such that every chain from the root of T to a node on C contains

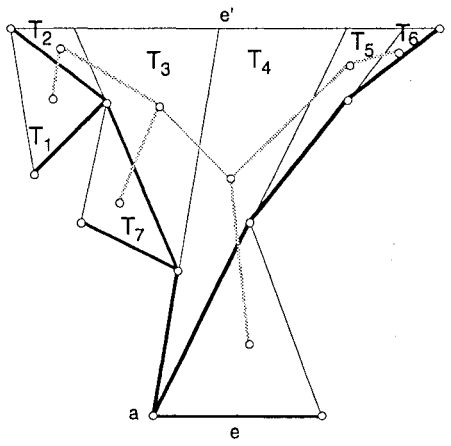

Figure 4.9: Leaves, funnels and the dual tree.

---

u.

Note: We will assume that the edges of $\overline{T}_a$ are oriented such that the triangle dual of the distal endpoint of an arc uv of T is to the left of the edge dual of uv. If this is not the case, we could do a reorientation in linear time.

We find a chain C on T by finding the chain of nodes of degree two adjacent to given a node η of degree one. Node η can be found in constant time if we have a depth-first search order on the nodes of T.

Note: Due to the nature of our triangulation, in this algorithm we will not assign the accumulated areas of the triangles determined by $\overline{T}_a$ in P_n directly to the edges of $\overline{T}_a$, instead we will "assign" these areas to the nodes of T. For example, $\mathcal{A}(\eta)$ will denote the area assigned to node η, which is the sum of the areas of the triangle dual of the nodes of a chain from η. We will use this information later to find the areas to the left of each link.

Let $\mathcal{A}(\Delta_\eta)$ denote the area of the triangle Δ_η dual of a node η of C.

Initially, we will assign $\mathcal{A}(\eta) = 0$ to each node η in C.

Let e'_η be the edge of P_n containing an edge of Δ_η. That is, if Δ_η is a leaf then e'_η is one of its edges, otherwise e'_η is the lid of the funnel containing Δ_η (for example e' corresponds to triangles T_2, T_3, T_4, T_5 and T_6 in Figure 4.9). Let y_η be the vertex of Δ_η not in e'_η. Let r_η be the edge of $\overline{T}_a$ leaving Δ_η to its left; in this way r_η is the *right edge of* Δ_η.

Let $\ell(e'_\eta)$ denote the link from edge e'_η to the base e of P_n, in the case that it does exist.

Algorithm 4.5.1 *To find links and distal areas*

1. *Find a chain of nodes* $C = \{\eta_1, \ldots, \eta_m\}$ *from a leaf* η_1 *on* T.
2. $\mathcal{A}(\eta_1) = \mathcal{A}(\Delta_{\eta_1})$
3. *If* $e'_{\eta_1} \in V(e)$ *then* $\ell(e'_{\eta_1}) = \chi(r_{\eta_1})$
4. *For* $i = 2$ *to* m, *do:*
   (a) $\mathcal{A}(\eta_i) = \mathcal{A}(\eta_{i-1}) + \mathcal{A}(\Delta_{\eta_i})$
   (b) *If* $e'_{\eta_i} \in V(e)$ *then* $\ell(e'_{\eta_i}) = \chi(r_{\eta_i})$
5. *Delete* C *from* T.
6. *Repeat until no more chains are left in* T.

Since there is a linear number of triangles, the number of nodes of T is also linear.

For each node η, Steps 4a and 4b take constant time, since $\chi(r_\eta)$ will always intersect e'_η and e. $\chi(r_\eta)$ intersects edge $e'_\eta = \overline{gd}$ because it is the lid

of a funnel; it also intersects the base $e = \overline{ab}$, because r_η is part of the shortest path from a to d but it cannot be part of CL_a^g since CL_v^g is the shortest path from v to g.

In order to calculate the area $\mathcal{A}(s)$ to the left of the segment s of $\overline{T}_a$ from the areas stored on the nodes of T we proceed as follows (initially we suppose $\mathcal{A}(s) = 0$, for all s):

Algorithm 4.5.2 *To find distal left areas*

1. *For each node η of T do:*

   $$\mathcal{A}(r_\eta) = \mathcal{A}(r_\eta) + \mathcal{A}(\eta)$$

This is because one edge of $\overline{T}_a$ may be the "right edge" of more than one triangle (in our case, two triangles). Here we are only assigning areas to those edges of $\overline{T}_a$ which determine links.

Therefore, we have:

Lemma 4.5.1 *We can find the area to the left of all the edges of $\overline{T}_a$ and the links for all visible edges of P_n in linear time.*

As we mentioned before, we still need to calculate the proximal extensions and the area to the left of the edges of T_a. Let ℓ be a link found by Algorithm 4.5.1 determined by the segment $\overline{uv}$ of T_a, with u being the proximal endpoint. We will create $\chi^P(s)$ (see Page 45), for each edge s in the chain CL_a^u. (see Figure 4.10).

Observe that all the extensions of a segment s in CL_a^u are on the same side of $\chi^P(\overline{uv})$. Furthermore, they all intersect e and they determine a partition into disjoint triangles of the region determined by CL_a^u, e and $\chi^P(\overline{uv})$; this is

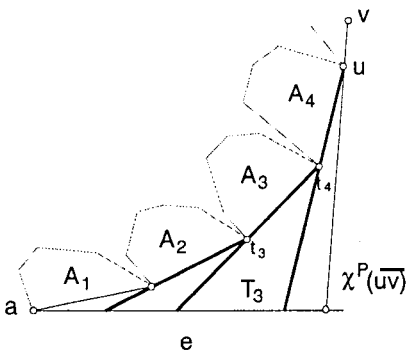

Figure 4.10: Proximal extensions.

---

because CL_a^u is a convex chain. Therefore $\chi^P(s)$ can be obtained in constant time and this allows us to obtain in linear time the area to the left of all our links as follows:

We will first assign areas to the vertices of T_a and not to edges, in order to simplify the algorithm.

Algorithm 4.5.3 *To find areas to the left*

Let $\overline{uv}$ be an edge of T_a defining a link.

Let $C = \{t_1 = a, \ldots, t_{m-1} = u, t_m = v\}$ be the shortest path from a to v in T_a.

1. $\mathcal{A}(t_1) = 0$

2. *For $i = 2$ to $m - 1$, do:*

   (a) *Let Δ_i be the triangle bounded by $\chi^P(\overline{t_{i-1}t_i})$, e and $\chi^P(\overline{t_it_{i+1}})$.*

   (b) $\mathcal{A}(t_i) = \mathcal{A}(t_{i-1}) + \mathcal{A}(\chi^D(t_i)) + \mathcal{A}(\Delta_i)$

For example in Figure 4.10, we have in Step 2b $\mathcal{A}(t_4) = \mathcal{A}(t_3) + \mathcal{A}(A_3) + \mathcal{A}(T_3)$.

Observe that with Algorithm 4.5.3 we can calculate the area to the left of the link $\ell = \chi(\overline{uv})$ by $\mathcal{A}(\ell) = \mathcal{A}(u) + \mathcal{A}(\chi^D(\overline{uv}))$.

Since we can find all the shortest paths from the root in linear time from T_a, we find the areas to the left of all the links in linear time, and we have:

Theorem 4.5.1 *Preprocessing a polygon P_n as in Problem 4.1.2 and finding the areas to the left of all its links can be done in linear time using linear space.*

4.6 Cuts

With the above preprocessing, let $c = \overline{pq}$ be a query cut. Suppose we find p in the base e and q in the edge e' (this can be done in logarithmic time with a binary search on the edges of $V(e)$). Then we can find the area $\mathcal{A}(c)$ to the left of c in constant time as follows:

Algorithm 4.6.1 *Cutting a simple polygon with a line segment*

1. *Let $\ell(e') = \overline{uv}$ be the link corresponding to edge e' found by Algorithm 4.5.1, with u on e and v on e'.*

2. *Let $\mathcal{A}(\ell(e'))$ be the area to the left of $\ell(e')$ also found by Algorithm 4.5.1.*

3. *Determine the intersection point r of c and $\ell(e')$. Since $\ell(e')$ is the support of the left and right chains from the base to e', point x always exists.*

4. *Let $\mathcal{A}(pur)$ and $\mathcal{A}(qvr)$ be the (positive) areas of the triangle with vertices $\{p, u, r\}$ and $\{q, v, r\}$ respectively.*

5. *If p is to the right of $\ell(e')$ then*

   $\mathcal{A}(c) = \mathcal{A}(\ell(e')) + \mathcal{A}(pur) - \mathcal{A}(qvr)$ *(see Figure 4.11a)*

   else

   $\mathcal{A}(c) = \mathcal{A}(\ell(e')) - \mathcal{A}(pur) + \mathcal{A}(qvr)$ *(see Figure 4.11b)*

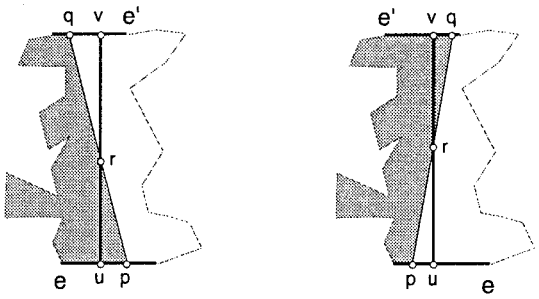

Figure 4.11: Two cases for intersections of cut $c = \overline{pq}$ with the link $\ell(e') = \overline{uv}$.

Since the intersections and the areas of the triangles can be found in constant time, we have:

Theorem 4.6.1 *With linear time preprocessing, Problem 4.1.2 can be solved in constant time given the edges of P_n intersected by the cut.*

Now, from Theorem 4.2.1, Equation 4.1 and Theorem 4.6.1 we have:

Theorem 4.6.2 *With $O(n \log n)$ preprocessing, we can find the area of a polygon cut by an interior line segment in logarithmic time.*

Before we show some extensions to our problem, we will mention a modification to the solution to Problem 4.1.2 to solve Problem 4.1.1 directly.

If we consider every edge of P_n as a base, we need linear time preprocessing (see Theorem 4.5.1) to find the area to the left of the links of the weakly visible edges from each base e of P_n. Assume that our cut c passes through the edges e and e' of P_n. We can determine in constant time the area of P_n cut by c (see Theorem 4.6.1). Therefore we have:

Theorem 4.6.3 *Provided a quadratic preprocessing and the edges of P_n cut by the line segment c, we can determine in constant time the area of P_n cut by c.*

4.7 Extensions

This is a short note on the extensions for the algorithm just presented.

We can observe that the concept of area was not necessary for the algorithms in Sections 4.5 and 4.6. Indeed the only two properties of $\mathcal{A}$ that we use are its additivity, and the fact that it can be defined for any interior triangle using only the values at its vertices (in this case, the vertex coordinates). Thus we can apply this algorithm to obtain other additive measures from cuts of polygons.

The *perimeter* $\mathcal{P}(P_n)$ of a polygon P_n is the sum of the lengths of its n edges. We have:

$$\mathcal{P}(A \cup B) = \mathcal{P}(A) + \mathcal{P}(B) - 2\mathcal{P}(A \cap B)$$

for two triangles A and B of a triangulation of P_n. Therefore, we can substitute "perimeter" for "area" in our algorithms.

Note that the perimeter of a polygon cut by an interior segment can be calculated in logarithmic time with linear preprocessing. The idea is to accumulate at each vertex v_i of P_n the length $L(i)$ of the (counterclockwise) polygonal chain $C(v_1, v_i)$ from a fixed vertex v_1 to v_i. Suppose that c cuts the edges $\overline{v_j v_{j+1}}$ at x and $\overline{v_k v_{k+1}}$ at y, with $j < k$. Then the perimeter of the portion of P_n not containing v_1 is

$$\mathcal{P}(P_n^-) = L(k) - L(j+1) + |v_{j+1} - x| + |v_k - y| + |x - y|$$

where $|v_{j+1} - x|$ and $|v_k - y|$ are the lengths of those portions of the edges cut which are included in P_n^- and $|x - y|$ is the length of the cut.

The *equibarycenter* of a set of points $P_n = \{v_1, \ldots, v_n\}$ with weights (or electrical charges) $\{p_1, \ldots, p_n\}$ is defined as the average:

$$E(P_n) = \frac{\sum_{i=1}^n p_i v_i}{\sum_{i=1}^n p_i}.$$

This value is clearly additive since

$$E(A_n \cup B_m) = \frac{\sum_{i=1}^n a_i E(A_n) + \sum_{i=1}^m b_i E(B_m)}{\sum_{i=1}^n a_i + \sum_{i=1}^m b_i}$$

where $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_m\}$ are the weights of the point sets A_n and B_m respectively.

The *centroid* (also known as *center of mass* or *center of gravity*) of a plate (in our case, with uniform unit density) represented by a simple polygon P_n with vertices $\{v_1, \ldots, v_n\}$, is defined as an average:

$$C(P_n) = \left(\frac{\iint x dA}{\mathcal{A}(P_n)}, \frac{\iint y dA}{\mathcal{A}(P_n)} \right)$$

where the integrals are defined over all the points of the polygon.

From physics we can observe that the weight (which in our case is the area) of a polygonal plate can be concentrated in its centroid. In this way, we can transform the problem of finding the centroid of a system of two polygons to the problem of finding the equibarycenter of their individual centroids. We also observe that the centroid of a triangle (with uniform unit density) and the equibarycenter of its vertices coincide at the intersection point of the three medians. The centroid is therefore additive, since:

$$C(A \cup B) = \frac{\mathcal{A}(A-B)C(A-B) + \mathcal{A}(A \cap B)C(A \cap B) + \mathcal{A}(B-A)C(B-A)}{\mathcal{A}(A \cup B)}$$

for any two polygons A and B. We can now apply our algorithms to calculate centroid instead of area.

Another extension comes from the solution of the following problem:

Problem 4.7.1 *Let P_n be a simple polygon and p be a point on its boundary. Calculate in $O(\log n)$ time, with $O(n \log n)$ preprocessing, the area cut by a segment s starting at p and having the direction of the vector $\vec{v}$ which points toward the interior of P_n.*

In order to transform this problem into an instance of Problem 4.1.1 in logarithmic time, we only need to find the second endpoint q of s in the boundary of P_n. If we start with the edge e containing p, we will be searching for the first edge e' of P_n (starting from p and in the direction of $\vec{s}$) intersected by the line ℓ containing s. The intersection point of ℓ with e' will be our second endpoint q. We can find q with a *ray shooting* algorithm by Chazelle et al (see [CEG+91]) running in logarithmic time and using a preprocessing of $O(n \log n)$.

4.8 Line and plane cuts

We will describe a generalizable procedure to find in linear time (a) the area of a polygon cut by a line and (b) the volume of a polyhedron cut by a plane. This procedure does not use preprocessing, therefore the problem of proving or disproving whether this problem can be sped up with preprocessing is open.

Note: The procedure to calculate areas in Sections 2.2 and 2.3 is called *the polar method* because of the "vectors" from the point p. We will describe here the *orthogonal projection* method. Both can be proved with the Green theorem or with the geometric procedure already given.

Problem 4.8.1 *Given a simple polygon P_n of n vertices in the plane and a query line ℓ, find the area $\mathcal{A}(P_n, \ell)$ of P_n to the left of ℓ.*

Let us give an orientation to ℓ (see Figure 4.12). In this way we define a left half plane ℓ^- and an order relation among the points on ℓ, i.e. $p_i < p_j$ iff ℓ^- is to the left of vector $\overrightarrow{p_i p_j}$.

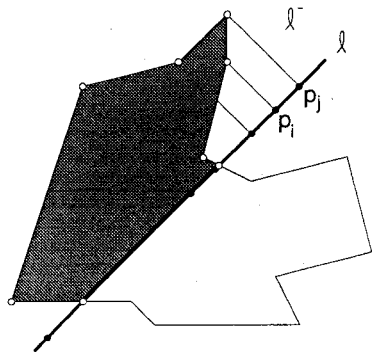

Figure 4.12: A polygon cut by a line.

The algorithm finds the polygon (or set of $O(n)$ disjoint polygons) $\ell(P_n)$ given by $P_n \cap \ell^-$, then orthogonally projects every vertex of $\ell(P_n)$ onto ℓ. Next, it calculates the signed area of the polygon determined by each edge of $\ell(P_n)$ and its orthogonal projection. Finally it adds together all these areas.

Algorithm 4.8.1 *Cutting a simple polygon with a line*

Let P_n be an n-gon with vertices $\{v_1, \ldots, v_n\}$ given in counterclockwise order, with $v_{n+1} = v_1$.

1. $\ell(P_n) = \emptyset$

2. *For $i = 1$ to n, do:*

   If v_i is to the left of ℓ then

   Add v_i to $\ell(P_n)$

   else

   If v_{i+1} is to the left of ℓ then

   Add to $\ell(P_n)$ the intersection point x, of $\overline{v_i v_{i+1}}$ with ℓ

   // Suppose $\ell(P_n)$ has m vertices $\{p_1, \ldots, p_m\}$, then:

3. *For $i = 1$ to m, do:*

   (a) *Let Q_i be the convex quadrilateral with vertices $\{p_i, p_{i+1}, p'_i, p'_{i+1}\}$ and area $a(Q_i)$.*

   Here p'_i is the orthogonal projection of p_i on ℓ.

   (b) *If $p'_i < p'_{i+1}$ then $A(Q_i) = -a(Q_i)$*

   else

   $A(Q_i) = a(Q_i)$

4. *The area $\mathcal{A}(P_n, \ell)$ of P_n to the left of ℓ is given by the formula:*

$$\mathcal{A}(P_n, \ell) = \sum_{i=1}^{m} A(Q_i)$$

Since the orientation (left or right of a point) on ℓ can be given in constant time, Step 2 can be done in linear time. Since the projections, comparisons and areas can be found in constant time, Steps 3 and 4 also take linear time. Therefore we have:

Theorem 4.8.1 *The area of a polygon to the left of a query line can be determined in linear time with no preprocessing.*

Similarly, we solve:

Problem 4.8.2 *Given a simple polyhedron P_n with n vertices in $\boldsymbol{R}^3$, and a query plane π, find the volume $\mathcal{V}(P_n, \pi)$ of P_n to the left of π.*

Give an orientation to π (see Figure 4.13). In this way we define a left half space π^- and a *sense of direction* among the points on π, i.e. $\langle p_i, p_j, p_k \rangle$ is a number with the value 1, 0 or -1, depending whether the normal vector of the plane generated by those three points (in the given order) is directed towards π^-, is on the plane or directed to the complement of π^- respectively.

The algorithm essentially finds the polyhedron (or set of disjoint polyhedra) $\pi(P_n)$ given $P_n \cap \pi^-$, then it orthogonally projects every face of $\pi(P_n)$ onto π. Next, it calculates the signed volume of the truncated prism determined by each face and its orthogonal projection. Finally, it adds together all these volumes.

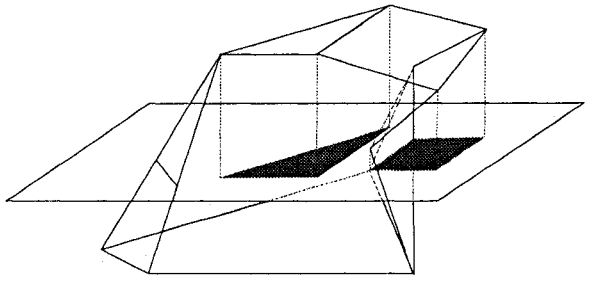

Figure 4.13: A polyhedron cut by a plane.

Algorithm 4.8.2 *Plane cut*

Let P_n be a polyhedron with n vertices and F_j one of its faces. Assume that F_j has j vertices given in counterclockwise order, i.e. any three consecutive vertices of F_j define a vector normal to F_j and directed out of the polyhedron.

1. $\pi(P_n) = \emptyset$
2. *For each face F_j of P_n, do:*
   (a) *Calculate $\ell(F_j)$, the portion of F_j to the left of π.*

   If we denote by ℓ^- the portion of the plane containing F_j and contained in π^-, then $\ell(F_j)$ is also $F_j \cap \ell^-$ which can be calculated in $O(j)$ time by Algorithm 4.8.1, where ℓ is the intersection of π with the plane containing F_j.

   (b) *If $\ell(F_j)$ is not empty, add it to $\pi(P_n)$*
3. *For each face F_j of $\pi(P_n)$, do:*
   (a) *Determine Q_j, the truncated prism determined by F_j and its orthogonal projection (or shadow) F_j' on π.*

(b) Calculate the positive volume $v(Q_j)$ of Q_j, with the formula:

$$v(Q_j) = l \cdot Q$$

If s is the segment having the centroids (see Page 57) of F_j and F'_j as its endpoints, then l is the length of s and Q is the area of a section of Q_j (also with j vertices) cut by a plane orthogonal to s.

(c) Let $V(Q_j)$ be the signed volume of Q_j given by:

$$V(F_j) = v(F_j) \cdot \langle p_i, p_{i+1}, p_{i+2} \rangle$$

for any three consecutive vertices p_i, p_{i+1}, p_{i+2} of F_j.

4. Calculate the volume $\mathcal{V}(P_n, \pi)$ to the left of π by:

$$\mathcal{V}(P_n, \pi) = \sum V(F_j)$$

where the sum goes through each face F_j in $\pi(P_n)$

4.9 Open problems

Here we include some open problems on cutting planar and 3-D figures.

Problem 4.9.1 *Linear and planar cuts of polytopes*

Without any constraint on time or space for preprocessing, is there a way to improve the time complexity for Problems 4.8.1 and 4.8.2?.

Problem 4.9.2 *Cutting the fan of wedges*

Let $F_n = \{w_1, \ldots, w_n\}$ be a set of n non-overlapping wedge regions in the plane, with common apex O and all lying to the right of the line l containing O (see Figure 4.14). Is there a way of preprocessing F_n such that the area $\mathcal{A}(F_n, \ell)$ of F_n to the left of the query line ℓ can be found in sublinear time?.

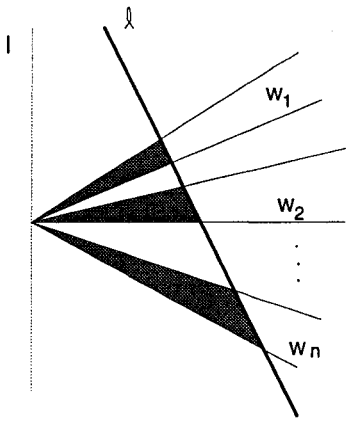

Figure 4.14: Cutting a fan.

The trivial values for the area are ∞ and 0, for the cases when ℓ does not intersect F_n or when ℓ is parallel to one edge of F_n or contains O. Otherwise, a set of n bounded triangles is determined by the wedges and ℓ. This problem can be solved in linear time without any preprocessing by simply adding together the areas of the n triangles determined by ℓ.

Problem 4.9.3 *Cutting the infinite pyramid*

Let P_n be the unbounded pyramidal region of $\boldsymbol{R}^3$ determined by n rays $\{r_1, \ldots, r_n\}$ originating at the apex O and above the plane Π containing O (see Figure 4.15). Assume, for example, that the rays are given in counterclockwise order of the vertices of a simple polygon B_n determined by the

intersections of P_n with a parallel plane above Π, and whose interior is inside P_n. Is there a way of preprocessing P_n such that the volume $\mathcal{V}(P_n, \pi)$ of P_n below a query plane π can be determined in sublinear time?

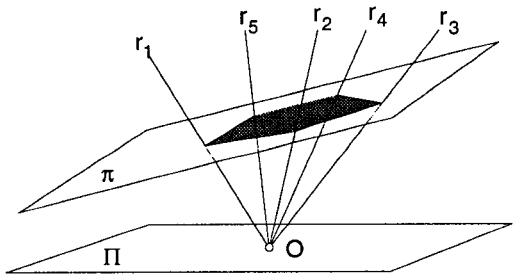

Figure 4.15: Cutting the infinite pyramid.

Again, the trivial solutions for the volume are ∞ and 0, for the cases when π does not intersect P_n or when π is parallel to a face or edge of P_n or contains O. Otherwise, we can observe a bounded pyramid having the polygon B_n as base. This problem can be solved in linear time without any preprocessing, by simply calculating the area of B_n and then multiplying it by 1/3 of the height of O on π (as in Equation 2.2).

Problem 4.9.4 *Lower bounds for preprocessing*

Show that the preprocessing time obtained in Theorem 4.6.2 and Theorem 4.6.3 is optimal or show lower bounds.

Theorem 4.6.2 says that we can solve Problem 4.1.1 in logarithmic time with $O(n \log n)$ preprocessing, while Theorem 4.6.3, playing the trade-off game, says that we can solve it in constant time, if the edges intersected by the cut are given and we have done an $O(n^2)$ preprocessing. While $O(n \log n)$

seems reasonable to be optimal, the quadratic preprocessing for the second case seems a bit excessive.

Chapter 5

Illumination of Stages

5.1 Introduction

Art Gallery and Illumination problems have been studied in the past two decades [O'R87], [She90]. The story starts with Victor Klee who in 1973 presented the following question: *How many guards are necessary and sufficient to patrol the art objects in an art gallery with n walls?*.

This question was an inspirational source for a long list of problems and published papers [Urr97a]. One of the best-known results in the area is Chvátal's Art Gallery Theorem which states that $\lfloor \frac{n}{3} \rfloor$ guards are sufficient and sometimes necessary to guard an art gallery represented by a simple polygon.

There exists a duality in the names used in the papers in this area; some of them refer to guards watching objects and others to light sources illuminating objects. From the abstract point of view, the concept is the same: We have two points p and q in a set S —for example a polygon— as in Figure 5.1,

and we say that point p *guards* or *illuminates* another point q if the segment $\overline{pq}$ is contained in the set S. An object P guards or illuminates an object Q if for every point $p \in P$ there exists a point $q \in Q$, such that p illuminates q. P weakly illuminates Q if there exists a $p \in P$ which illuminates a $q \in Q$.

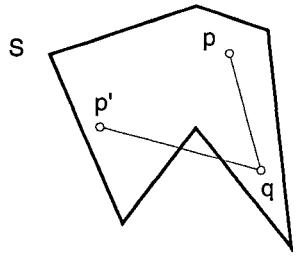

Figure 5.1: In polygon S, point p illuminates q and p' does not illuminate q.

A variety of constraints have been studied, defining objects such as point guards, vertex guards, edge guards, and mobile guards, as well as vertex and point floodlights. For example, in Chvátal's theorem we use *vertex guards*, that is, the guards are placed at the vertices of the polygon and they are not allowed to move. Of course, they can see in any direction around them —and we suppose they have outstanding vision to see as far as we want! The equivalent with lights are bulbs illuminating in every direction with unlimited power.

For our problem, we will be working with *point floodlights*, that is, light sources with a restricted aperture, such that the light rays emanate from the source called an *apex* illuminating inside an infinite angular region of the plane, sometimes called a *wedge*. The apexes of our floodlights can only be located in the given point set.

Floodlight Illumination Problems (see [O'R] and [Urr97a]) have enjoyed

a growing interest during the last ten years. Among the open problems in this area, the now classical Problem of Illumination of Stages was stated by Urrutia at [BGL+93] (see Figure 5.2).

Figure 5.2: The Problem of Illumination of Stages.

Problem 5.1.1 (Illumination of Stages) *Given a line segment s representing a stage, and a collection of n floodlight apexes and apertures, decide whether it is possible to illuminate s by rotating the floodlights around their apexes.*

It is worth noting that a light technician in the theater can decide, after trying for a while, if a given set of floodlights suffices to illuminate the stage of a theater or not. However, in the computer world only a brute force exponential procedure can be used and the complexity of the problem is still unknown.

The same technician would not be able to handle the following problem, however the resulting computational solution is not complicated.

Problem 5.1.2 *Given a stage, represented by the line segment s and a set of n points P on the same side of s, find a set of floodlights with apexes at P whose sum of apertures is the smallest, such that s belongs to their union.*

Here multiple floodlights are allowed per position, but floodlights with aperture zero are not admitted. The problem where the floodlights are restricted to only one per point in P remains open.

Notice that Problem 5.1.2 was solved in [CRCU93], with an optimal time of $\Theta(n \log n)$. However, during this thesis we found an accelerated solution when the points are vertices of convex polygons. Our algorithm for this case now takes linear time. We also produced a companion videotape for the solution given in [CRCU93], which was presented at [CCRCU98].

5.2 Two floodlight sources

In order to introduce the problem, we first present the two floodlight sources case. That is:

Problem 5.2.1 *Given two sources for floodlights at the points p and q above a stage represented by a horizontal line segment s, illuminate s with floodlights having apexes at p and q using the minimum sum for their angles.*

We will say that a point z on the stage is illuminated from the point p (or that p illuminates z) if z is illuminated from a floodlight with its apex placed at p.

We notice that some portions of the stage are illuminated better from one of the floodlights than from the other. For example, consider a segment $\overline{xy}$ on s as in Figure 5.3a. Consider the circle C passing through x, y and p. We know from elementary geometry that there are three possibilities for point q, namely q can be inside, on or outside C. The angle $\angle xpy$ of the floodlight

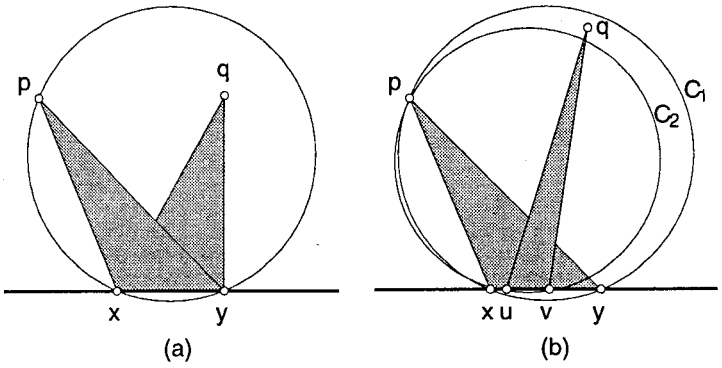

Figure 5.3: (a) Illuminating a segment on s. (b) Is p the best choice to illuminate *any* point on segment $\overline{xy}$?

---

from p will be smaller, equal or larger respectively, than the angle $\angle xqy$ of the floodlight from q. Therefore we say, respectively, that p is better than q, that p and q are equally good or that q is better than p to illuminate the segment $\overline{xy}$.

However, given the above observation, one may think that in order to illuminate s with the minimum sum of angles, we can take any partition of s into segments and choose the best point, either p or q, to illuminate them. This is not true, that is, the best choice to illuminate a segment is not necessarily the best choice to illuminate each single point on it, as we can observe in Figure 5.3b. Here, we need a definition:

Definition 5.2.1 *We say that point p is better than point q to illuminate a single point z on the stage, if there exists $\delta > 0$ such that for every $\varepsilon < \delta$ with ε positive, p is better than q to illuminate a segment t of length ε on the stage containing z. If no such δ exists, we say that both points are equally good to illuminate z.*

For example, in Figure 5.3b point p is better than point q to illuminate the segment $\overline{xy}$ since q is inside the circle C_1. However, q is outside the circle C_2 passing through points u and v on the stage, therefore q is better than p to illuminate the segment $\overline{uv}$, which is contained in $\overline{xy}$. Therefore we need a criterion to decide which point is the best choice to place a floodlight to illuminate a single point on the stage.

Let C_L and C_R be the circles passing through p and q and tangent to the stage at the points z_L and z_R respectively. Assume P is in the clockwise arc $z_R q$ as in Figure 5.4.

Let δ be the minimum distance between z and both z_L and z_R. Then for every $\varepsilon < \delta$, we construct the open segment $\overline{xy}$ of length ε containing z and the circle C passing through x, y and p. We claim that our choice for the best point to illuminate any segment $\overline{xy}$ of length ε does not change for any $\varepsilon < \delta$.

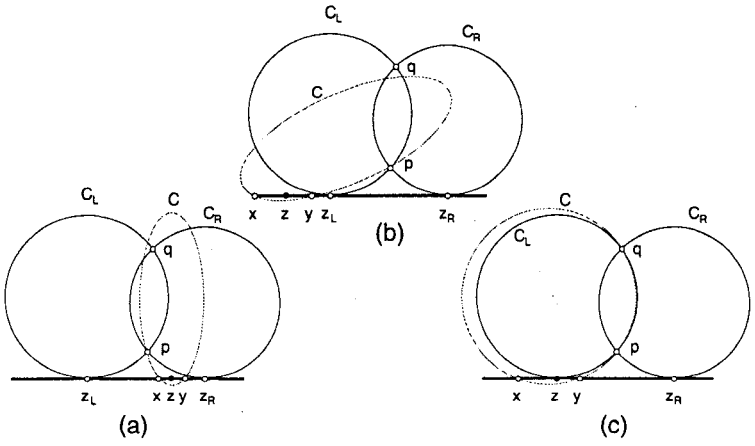

Figure 5.4: Best illumination for a point z on the stage.

Let $\varepsilon < \delta$. There are three cases to consider:

If z is inside $\overline{z_L z_R}$ (see Figure 5.4a), then p cannot illuminate a segment $\overline{xy}$ of length ε. The segment will always be contained in $\overline{z_L z_R}$ and a circle C passing through x, y and p leaves q outside, otherwise C would intersect C_R four times.

The second case is where z lies outside $\overline{z_L z_R}$ (see Figure 5.4b). Then q cannot illuminate a segment $\overline{xy}$ of length ε. In this case the segment $\overline{xy}$ is always outside $\overline{z_L z_R}$ and therefore the circle C passing through x, y and p always contains q, otherwise C would intersect C_L four times.

The third case is when z coincides with z_L or z_R (see Figure 5.4c). In this case p, q and z are co-circular and we will show that for any $\delta > 0$, we can always find two segments of length $\varepsilon < \delta$ such that one is better illuminated from p and the other is better illuminated from q.

Indeed, let $\delta > 0$ and select the circle C' determining a segment $\overline{xy}$ of length $\varepsilon < \delta/2$ on the stage from the family F of circles passing through p and q. Since C' belongs to F, it will contain the point z, therefore z belongs to $\overline{xy}$. Now take the circle C^{-h} passing through $\{p, x-h, y-h\}$ and the circle C^{+h} passing through $\{p, x+h, y+h\}$, where h is half of the minimum distance between z and both x and y. Only one of C^{-h} and C^{+h} contains q. Suppose C^{-h} contains q but C^{+h} does not. Then $\overline{(x-h)(y-h)}$ is better illuminated from p than from q but $\overline{(x+h)(y+h)}$ is better illuminated from q than from p. Furthermore both segments, $\overline{(x-h)(y-h)}$ and $\overline{(x+h)(y+h)}$, contain z and have length ε.

Thus for the case when z is z_L or z_R, both p and q are equally good to illuminate z. We have:

Lemma 5.2.1 *Let p and q be two points above the stage s, and let z be a point on the stage. We say that q is better than p to illuminate z if it is outside the circle C passing through p and tangent to s at the point z. If q is interior to C then p is better than q and if z, p and q are co-circular, then both p and q are equally good to illuminate z.*

We call the points z_L and z_R *event points.* As an immediate result, we have (see Figure 5.5):

Lemma 5.2.2 *The region between z_L and z_R is better illuminated from q than from p if p is in the clockwise arc $z_R q$. The rest of the stage is better illuminated from p than from q.*

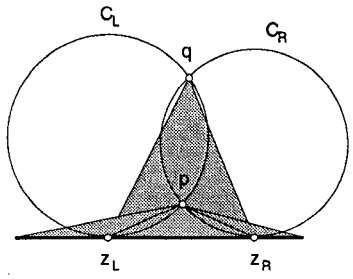

Figure 5.5: Illumination of a stage with two floodlights whose sum of angles is minimum.

Lemma 5.2.1 gives a characterization to find the best light position to illuminate a point on the stage which we will use in the next section. On the other hand, Lemma 5.2.2 provides us with the solution for our problem in the case of two floodlights. As the tangent circles can be found in constant time with a construction by Apollonius (see for example [Eve72]), we have:

Theorem 5.2.1 *A stage can be illuminated from two floodlight sources using the minimum sum for their angles with the above procedure in constant time.*

5.3 The general case

Let $P = \{p_1, \ldots, p_n\}$ be a set of floodlight sources above the horizontal line S representing the stage.

Consider the family of circles F_z tangent to a fixed point z on the stage and containing P. Let $p_i \in P$ be one point of those intersected by the smallest member C of F_z. Since all other points of P are contained in C, we have that p_i is the best place to put a floodlight illuminating z. If there was another point p_j of P intersected by C, then both points p_i and p_j would be equally good to illuminate z. This is a direct result of Lemma 5.2.1 (see Figure 5.6).

Lemma 5.3.1 *The best position for a floodlight to illuminate a point z on the stage s from the set of floodlight sources P, is the point p_i intersected by the smallest circle containing P and tangent to s at point z.*

The *convex hull* H_P of a set of points P is the minimum convex set containing it. Observe that if a point p_j inside H_P illuminates a point z of the stage, z is still better illuminated from a point p_i intersected by the smallest member of F_z. The point p_i is a vertex of H_P, otherwise the lines $\overline{p_i p_k}$ and $\overline{p_i p_{k-1}}$ would separate points of P, p_{k-1} and p_k being the two neighbor vertices of p_i in H_P. If that were the case, a circle of F_z passing through p_i would not contain all the points of P. Therefore we have:

Lemma 5.3.2 *Only vertices of the convex hull of P can be candidates to illuminate points on the stage with floodlights of minimum sum of angles.*

For our further analysis, we can assume that the points of P are the vertices of a convex polygon.

A circle tangent to the stage and passing through two points p_i and p_j of P is called a *critical circle* if it contains all other points of P. In such a case, the tangency point will be called *event point.*

Observation 5.3.1 *Suppose that p_i lies on the clockwise arc zp_j of C, as in Figure 5.6b. Observe that if y is a point to the right of z, and x a point to the left of z, a circle passing through p_i, z and y contains p_j, and a circle passing through p_i, x and z does not contain p_j. Therefore the best point to illuminate a neighborhood to the right of z is p_i, and the best point to illuminate a neighborhood to the left of z is p_j.*

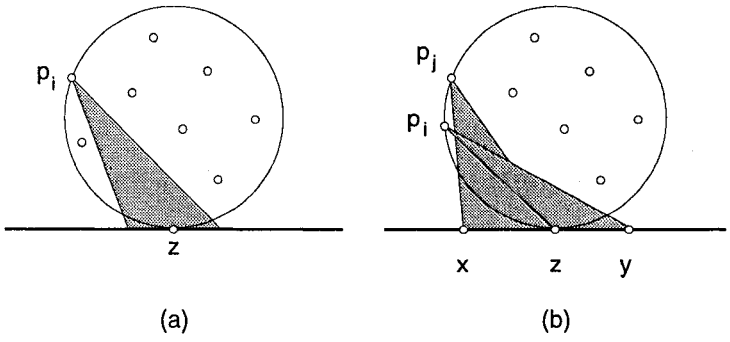

Figure 5.6: (a) Choosing the best point to illuminate z. (b) p_i illuminates to the right of z and p_j illuminates to the left of z.

This observation suggests a partition of the stage given by the event points. It also directs us to consider the points of P in pairs in a cyclical order,

in order to determine all the event points on S. For example, if we obtain all the event points from the rightmost to the leftmost we obtain a cyclical order for the points of P, in this case, clockwise. Finally, Observation 5.3.1 tells us how to determine which floodlight source illuminates what portion of the stage S optimally.

Let p_1 be the point of P nearest the stage and suppose that the points of P are labeled $\{p_1, \ldots, p_n\}$ in clockwise order. Let F_1 be the family of circles tangent to S which pass through the point p_1 and each of the other points of P.

Let C_L be the circle of F_1 defining the leftmost tangency point x on the stage. In the same way, let C_R be the circle of F_1, defining the rightmost tangency point y on the stage as shown in Figure 5.7a. Then P is contained in both circles, otherwise if point p_j of P were outside C_L or C_R (see Figure 5.7b), both tangency points z_1 and z_2 of the circles C_1 and C_2 passing through p_1 and p_j, would be outside $\overline{xy}$, otherwise C_1 or C_2 would have four intersections with C_L or C_R. Using this and Observation 5.3.1, we have:

Lemma 5.3.3 *The point of P nearest the stage S is always the best to illuminate the intervals $(-\infty, x]$ and $[y, +\infty)$ of the stage.*

Now our algorithm has a more defined shape:

Let $P = \{p_1, \ldots, p_n\}$ be a set of n points above the horizontal line S. Let $x = -\infty$; $y = \infty$ be the endpoints of the stage. Suppose C_{ij} and C'_{ij} are the circles passing through points p_i and p_j and tangent to S at points z_{ij} and z'_{ij}.

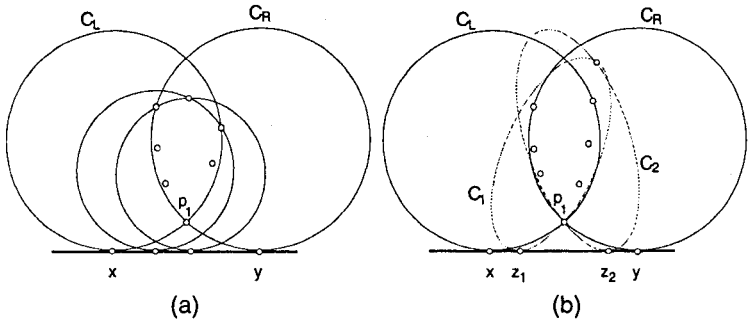

Figure 5.7: Left- and rightmost event points x and y respectively.

Algorithm 5.3.1 (FLIP) *Floodlight Illumination Problem*

// Preprocess:

1. *Calculate the convex hull of P relabeling the resulting vertices as $P = \{p_1, \ldots, p_m, p_{m+1} = p_1\}$, with p_1 being the closest vertex to S.*

2. *Calculate V_P, the FSVD of P.*

// Finding critical circles and event points (process):

1. $i = 1$; $z = y$

2. *While $i \leq m$ do:*

   (a) $j = i + 1$

   // Discard points not generating critical circles:

   (b) *While $P \not\subset C_{ij}$, $P \not\subset C'_{ij}$ and $j \leq m$ do:*

   $j = j + 1$

   (c) *If $j \leq m + 1$ then*

i. If $P \subset C_{ij}$ *then* $t_{ij} = z_{ij}$ *else* $t_{ij} = z'_{ij}$

ii. Illuminate the segment $\overline{t_{ij}z}$ *from* p_i

iii. $z = t_{ij}$ *;* $i = j$

3. Illuminate the segment $\overline{xz}$ *from* p_1

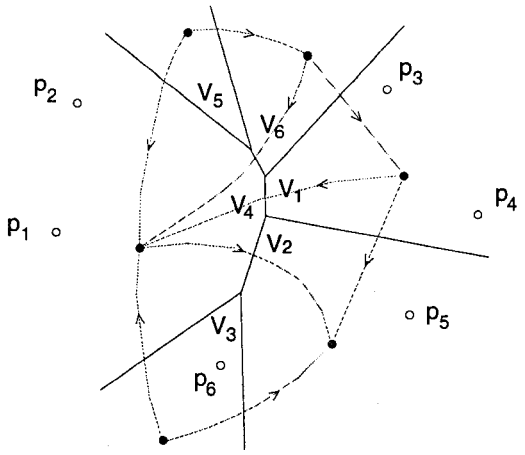

Figure 5.8: The FSVD of 6 points in convex position and its dual digraph.

In order to determine whether a circle C passing through the points p and q contains the set P, we first construct V_P, the *Furthest Site Voronoi Diagram* of P.

The Furthest Site Voronoi Diagram (FSVD) defines a partition of the plane into convex polygons V_i, associating the —possibly unbounded— polygon V_i with the point p_i of P (see Figure 5.8). V_i consists of the points further away from p_i than from any other point p_j of P. An edge shared by the polygons V_i and V_j of V_P is part of the bisector of p_i and p_j. The FSVD can be constructed in $O(n \log n)$ time in the general case, or in linear time if, as in our case, the points are vertices of a convex polygon [GO97].

We observe that the edges of V_P define a binary tree if the points of P are vertices of a convex polygon. A point on the boundary common to V_i and V_j is equidistant from p_i and p_j.

We wish to determine if the center c of the circle C passing through p_i and p_j is interior to a polygon V_k or not. In this way, if c is interior to V_k, it means that the distance between c and p_k is longer than the distance from c to p_i, which implies that p_k is outside the circle C and z_{ij} is not an event point (see Figure 5.9).

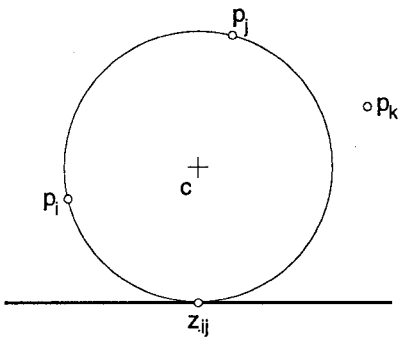

Figure 5.9: z_{ij} is not an event point because p_k is outside the circle and p_j will be discarded.

We will check in constant time whether c is on an edge of the FSVD or not. Our problem is now:

Problem 5.3.1 *Given the FSVD V_P of the vertices of a convex polygon $P = \{p_1, \ldots, p_n\}$, describe the preprocessing needed to determine in constant time if the center c of a circle C passing through p_i and p_j is on an edge of V_P.*

Construct the dual graph D of V_P, that is, take a node d_i per polygon V_i and an arc between two nodes d_i and d_j if the polygons V_i and V_j are

adjacent. D is a planar graph dual of the tree V_P, for which we always can find a node of degree 2. Now transform D into a digraph as follows: Let d_i be a node of degree 2 in D. Direct the two arcs adjacent to d_i outwards and temporarily delete d_i. At the same time, decrease the degree of the adjacent nodes and repeat the procedure, since the resulting graph is the dual of another tree.

Observe that the *outdegree* of each node (maximum number of edges departing from a node) of D is at most 2. With each arc between two nodes d_i and d_j of D, we store the corresponding edge e_{ij} of V_P.

With this preprocessing, each time we want to decide if the center c of the circle passing through points p_i and p_j intersects an edge (perhaps a ray) of V_P or not, we use our digraph D as follows: We determine first if V_i and V_j are adjacent, since if they are not, the bisector b of p_i and p_j is not part of an edge of V_P; that is, a point c' on b is inside a polygon V_k; in particular, the center c of C can be discarded at this step. We do this by verifying if one of the (at most) two arcs going out of d_i is directed towards d_j or if one of the (at most) two arcs going out of d_j is directed towards d_i. This takes at most four verifications. If V_i and V_j are found to be adjacent, we check whether c intersects e_{ij} or not, also in constant time.

The convex hull H_P of P can be obtained in $O(n \log n)$ time with standard techniques. Observe that this step is not necessary if the points are already the vertices of a convex polygon. The furthest site Voronoi diagram V_P of the vertices of a convex polygon can be constructed in linear time. The dual graph D_P takes linear time to construct, since there is a linear number of polygons in the planar graph V_P. Directing the arcs of D_P also takes

linear time. This can be done by performing a depth-first search on V_P, since the node of degree two that we need for our labeling is the dual of the region between two adjacent leaves of V_P. The preprocessing time is therefore $O(n \log n)$.

In order to count the number of iterations in our processing steps (Steps 1 to 3), observe that for a point p_j of P, either p_j is discarded or an event point is generated. Therefore the number of iterations in processing time is linear in m and the overall complexity of this algorithm is $O(n \log n)$.

Theorem 5.3.1 *The upper bound for the complexity of Problem 5.1.2 is $O(n \log n)$ in general or $O(n)$ if we assume that the points are the vertices of a convex polygon and ordered cyclically.*

5.4 The lower bound

We will find that Algorithm 5.3.1 is optimal. This can be observed after we sort a set of real numbers with it; that is, we are going to transform our problem into a sorting problem in linear time. As sorting takes $\Theta(n \log n)$ time, our algorithm to illuminate stages must take at least that time. Of course, if we know that the points for the illumination problem are already ordered, the lower bound is linear.

Let $\{x_1, \ldots, x_n\}$ be a set of real numbers, d the maximum distance between two of them, C a circle with diameter d and above the stage S, and S the real line $x = 0$. Find the real values $\{y_1, \ldots, y_n\}$, such that the points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ lie in the upper half of the boundary of C, as shown in Figure 5.10. Let P be this set of points and illuminate S from P using

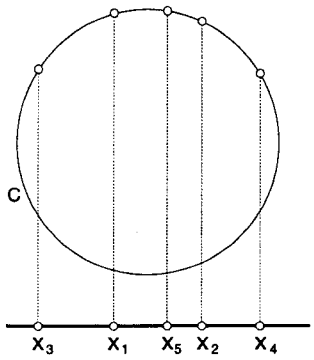

Figure 5.10: Finding a lower bound.

Algorithm 5.3.1, finding event points as described above. Due to Observation 5.3.1, an algorithm illuminating the stage will produce an order for the event points. This order induces an order on the points of P generating the event points and this in turn induces an order —reversed— of the real numbers given as data. Therefore, we have:

Theorem 5.4.1 *An algorithm to illuminate a stage with a collection of floodlights whose sum of angles is minimum, needs* $\Omega(n \log n)$ *time.*

Theorems 5.3.1 and 5.4.1 imply:

Theorem 5.4.2 *A stage can be illuminated from a set of points using floodlights minimizing the sum of their angles in optimal time* $\Theta(n \log n)$.

5.5 Notes

Here we have shown only the solution when the stage is a full line. From this solution we can derive the solution for a line segment stage as follows:

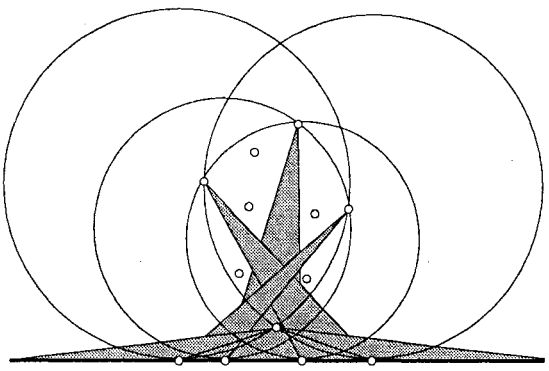

Figure 5.11: Illumination of a stage.

We consider in turn each of the floodlights chosen by the algorithm. If a floodlight from p_i illuminates outside the stage, then we discard p_i from consideration. If the floodlight from p_i illuminates only part of the stage, we reduce its angle until it illuminates up to the boundaries of the segment.

A note for degenerate cases: if we have two equal circles determined by three points, p_i, p_j and p_k of P, and $i \leq j \leq k$, we can discard p_j from consideration and illuminate the portion to the right of z_{ik} from p_i and the portion to its left from p_k.

Finally, if no circle is generated by p_i and p_j, that is, when $\overline{p_i p_j}$ is perpendicular to the stage, we take the base of the perpendicular $\overline{p_i p_j}$ on the stage as z_{ij}.

5.6 About the video

(...or the small guide for/from a beginner video-maker).

As mentioned earlier, the solution for the problem presented in this chap-

ter was implemented in a video demonstration. It was recorded in *real time mode* from a *multimedia* presentation, that is, the presentation was timed and then recorded in one shot. The presentation was entirely produced with the aid of Microsoft's PowerPoint version 7. The final video, with a duration of 7′30″, was recorded on VHS videotape.

PowerPoint was not designed for the purpose of producing videos, but rather as a tool for creating multimedia presentations. The differences arise, for example, when we want to control the time for recording or when we want an image in the computer "compatible" with a TV signal. Most of our animation is simply a sequence of fixed images displayed every certain predetermined interval (measured in fractions of a second) and PowerPoint does not have an accurate control for that (at least up to Version 7). We know of other programs such as *Director*, perhaps more convenient for this purpose but unfortunately not available at the time.

This is a list, not necessarily complete, of some factors to consider in producing a video:

The resolution —dots per square inch— used by a monitor and that used by a TV signal are different. Therefore some details such as making lines thicker, figures and texts with high contrast of colors and the use of "compatible" colors, have to be considered and adjusted early in the production and compared against a TV output.

PowerPoint draws each geometric figure on the screen and has to do so quickly enough to produce the impression of continuity of movement in our brain, but slow enough to allow our eyes to distinguish the shapes.

Unfortunately the program provides little control of that aspect. Therefore, in order to produce acceptable results, we depend on the speed of the processor ($\geq$200MHz) and on some tricks such as for example, drawing invisible figures to slow down the animation. The duration of the video was required to be $\leq$ 10 minutes for its presentation in the conference.

For the draft version we used a built-in multimedia card, however for the final version we used a *Real Time Converter for video (RTC).* The RTC has an input from the monitor port of the PC and an output to each of a BetaCAM and a VHS professional recorders (we used the BetaCAM as a *master tape* for personal copies and we sent the VHS to the conference).

The audio track (voice and music) is maybe the most noticeable part of the video. Therefore it has to be recorded with the best possible quality, since it will be degraded during the process of changing it from one medium to another. Needless to say, our tests with a normal *open-air* computer microphone had unsatisfactory results, but they were good enough to measure the preliminary durations. For the final version, in our case, a professional narrator was hired and the recording made at a university studio. Then the audio track was digitalized and manipulated with the *Sound Effects* audio editor for the MacIntosh and then synchronized to the video pieces with the *CoolEdit* audio editor for the PC (both programs are available on the Net as shareware).

An interesting detail in this process was that due to a small mistake in

the script, the narrator switched the spelling of two letters, but aided by digital editing of the sound, essentially a careful *cut and paste* process, the correction was unnoticeable to the ear.

The video was submitted and accepted for presentation in the video review of a conference [CCRCU98].

Chapter 6

Conclusion

The following table lists the complexity results for the main algorithms presented in this thesis:

Algorithm	**Convex case /preprocess**	**General case /preprocess**
Area of a polygon cut by a segment	$O(1)/O(n)$ (segment through two vertices)	$O(\log n)/O(n \log n)$ $O(1)/O(n^2)$
cut by a line	$O(\log n)/O(n)$ (any line)	$O(n)/-$
Volume of a polyhedron cut by a plane	$O(n)/-$	$O(n)/-$
Illuminating a stage	$O(n)/O(n)$	$O(n)/O(n \log n)$

Bibliography

[All86] Eugene L. Allgower. Computing volumes of polyhedra. *Mathematics of Computation*, 46(173):171–174, 1986.

[BGL+93] Prosenjit Bose, Leonidas Guibas, Anna Lubiw, Mark Overmars, Diane Souvaine, and Jorge Urrutia. The floodlight problem. In *Proceedings of the 5th Canadian Conference on Computational Geometry*, pages 399–404, Waterloo, ON, Canada, August 1993. University of Waterloo.

[CCAU98] Jurek Czyzowicz, Felipe Contreras-Alcalá, and Jorge Urrutia. On measuring areas of polygons. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, page (to apear), Montréal, QC, Canada, August 1998. McGill University.

[CCRCU98] Felipe Contreras, Jurek Czyzowicz, Eduardo Rivera-Campo, and Jorge Urrutia. Optimal floodlight illumination of stages (video review). In *Proceedings of the 14th ACM Symposium on Computational Geometry*, Minneapolis, Minnesota, USA, June 1998. ACM.

[CEG+91] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. In *Proc. 18th Internat. Colloq. Automata Lang. Program.*, volume 510 of *Lecture Notes in Computer Science*, pages 661–673. Springer-Verlag, 1991.

[CG67] H. S. M. Coxeter and S. L. Greitzer. *Geometry Revisited.* Mathematical Association of America, Washington, DC, 1967.

[Cha82] B. Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 339–349, 1982.

[Cha91] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6:485–524, 1991.

[CRCU93] Jurek Czyzowicz, Eduardo Rivera-Campo, and Jorge Urrutia. Optimal floodlight illumination of stages. In *Proceedings of the 5th Canadian Conference on Computational Geometry*, pages 393–398, Waterloo, ON, Canada, August 1993. University of Waterloo.

[DO90] M. Díaz and J. O'Rourke. Ham-sadwich sectioning of polygons. In *Proceedings of the 2th Canadian Conference on Computational Geometry*, pages 282–286, Ottawa, ON, Canada, August 1990. University of Ottawa.

[DR95] Ricardo Diaz and Sinai Robins. Pick's formula via the weierstrass $\wp$-function. *The American Mathematical Monthly*, 102(5):431–437, 1995.

[ECU95] V. Estivill-Castro and J. Urrutia. Two-floodlight illumination of convex polygons. In *Proc. 4th Workshop Algorithms Data Struct.*, volume 955 of *Lecture Notes in Computer Science*, pages 62–73. Springer-Verlag, 1995.

[Eve72] Howard Whitley Eves. *A survey of geometry.* Boston: Allyn and Bacon, 1972.

[Fis78] S. Fisk. A short proof of Chvátal's watchman theorem. *J. Combin. Theory Ser. B*, 24:374, 1978.

[GMP+93] S. K. Ghosh, A. Maheshwari, S. P. Pal, S. Saluja, and C. E. Veni Madhavan. Characterizing and recognizing weak visibility polygons. *Comput. Geom. Theory Appl.*, 3:213–233, 1993.

[GO97] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC Press LLC, Boca Raton, FL, 1997.

[GS93] Branko Grünbaum and G. C. Shephard. Pick's theorem. *The American Mathematical Monthly*, 100(2):150–161, 1993.

[KKK83] J. Kahn, M. M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM J. Algebraic Discrete Methods*, 4:194–206, 1983.

[Lig88] James A. Liggett. Exact formulae for areas, volumes and moments of polygons and polyhedra. *Comm. Appl. Numer. Methods*, 4(6):815–820, 1988.

[MSW93] David M. Mount, Ruth Silverman, and Angela Wu. On the area of overlap of translated polygons. *Vision geometry*, II:254–264, 1993.

[Mun75] James R. Munkres. *Topology: a first course.* Prentice-Hall Inc., Englewood Cliffs, NJ, 1975.

[O'R] J. O'Rourke. Discrete and computational geometry: Ten years later. *to appear*, .

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[OSS95] Joseph O'Rourke, Thomas Shermer, and Ileana Streinu. Illuminating convex polygons with vertex floodlights. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 151–156, 1995.

[PS85] F. P. Preparata and M. Ian Shamos. *Computational Geometry, An Introduction.* Texts and Monographs in Computer Science. Springer-Verlag, New York – Berlin, 1985.

[Rob95] David P. Robbins. Areas of polygons inscribed in a circle. *Amer. Math. Monthly*, 102(6):523–530, 1995.

[She90] Thomas Shermer. Recent results in art galleries. Technical Report 90-10, School of Computing Science, Simon Fraser University, October 1990.

[She92] T. C. Shermer. A linear time algorithm for bisecting a polygon. *Inform. Process. Lett.*, 41(1):135–140, 1992.

[Ski91] Steven S. Skiena. Probing convex polygons with half-planes. *J. Algorithms*, 12(3):359–374, 1991.

[Spe86] Ted Speevak. An efficient algorithm for obtaining the volume of a special kind of pyramid and application to convex polyhedra. *Mathematics of Computation*, 46(174):531–536, 1986.

[Ste69] H. Steinhaus. *Mathematical Snapshots.* Oxford University Press, 1969.

[Sto91] Ivan Stojmenović. Bisections and ham-sandwich cuts of convex polygons and polyhedra. *Inform. Process. Lett.*, 38(1):15–21, 1991.

[Str93] Gilbert Strang. Polar area is the average of strip areas. *American Mathematical Monthly*, 100(3):250–254, 1993.

[Urr97a] Jorge Urrutia. Art gallery and illumination problems. In J.R. Sack and J. Urrutia, editors, *Handbook on Computational Geometry*, pages 387–434. Elsevier Science Publishers, 1997.

[Urr97b] Jorge Urrutia. Sixth proof of the orthogonal art gallery theorem. Technical Report TR-97-03, University of Ottawa, February 1997.

[Ven95] R. Venkatasubramanian. On the area of intersection between two closed 2-d objects. *Inform. Sci.*, 82(1–2):25–44, 1995.

Index