INFORMATION TO USERS

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

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

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

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

Visibility Problems Occurring in Radiation Treatment Planning

By

Jian Huang, B.Sc.

*A thesis submitted to the
Faculty of Graduate Studies and Research
in Partial fulfillment of the requirements for
the degree of
Master of Computer Science*

*Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario
April 28, 2001*

© Copyright
2001, Jian Huang

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-61012-8

Canada

The undersigned recommend to the Faculty of Graduate Studies

and Research acceptance of the thesis

"Visibility Problems Occurring in Radiation Treatment Planning"

submitted by Jian Huang, B.Sc.,

in partial fulfillment of the requirements for

the degree of Master of Computer Science

Handwritten signature of Jay-E Sack, Thesis Supervisor, written over a horizontal line.

Thesis Supervisor

Handwritten signature of the Chairman, School Computer Science, written over a horizontal line.

Chairman, School Computer Science

Carleton University

Date

Abstract

We consider geometric problems arising in the treatment planning of a radiation therapy for cancer patients. The goal of radiation therapy is to maximize the high energy x-rays to the cancer cells while minimizing the risk of doing damage to the surrounding tissues or normal organs. We model these problems as visibility problems in which the goal is to find a (set of) point(s) from which a tumor polygon T is wholly visible in the presence of a set of normal organ polygons.

Acknowledgements

I wish to express my gratitude to the many people who have helped make this thesis a reality. Firstly is my supervisor, Professor Sack, to whom I am forever grateful for his patience, his guidance and his advice. Secondly, a special thanks goes to Patrick Morin for many discussions and ideas in developing the algorithms. Thirdly, I would like to thank Doron Nussbaum for reading the draft of the thesis and giving his constructive criticism. In addition, I would also like to thank my friend Vincent Young for helping me in putting the final package together. And finally, I would like to thank my family and my husband, Fei Tan, for his support, and the personal care of our two young children, Aaron and our new arrival Yvonne.

Table of Contents

ACCEPTANCE SHEET	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vii

Chapter 1: Introduction

1.1 Motivation	1
1.2 Definitions	4
1.3 Scope of the Thesis	6

Chapter 2: Visibility Polygon Research

2.1 Introduction	7
2.2 Visibility Region from a Point	10
2.3 Visibility Region from a Line Segment	12
2.4 Visibility Region from a Polygon	14

Chapter 3: Algorithm for Computing the Visibility Polygon from a Polygon inside a Simple Polygon

3.1 Specification	15
3.2 Definitions and Notations	16
3.3 Introduction	17
3.4 Description	20
3.5 Running-time Analysis	23
3.6 Discussion	24

Chapter 4: The Art Gallery Problem

4.1 Introduction	25
4.2 Basic Concepts	26
4.3 Guarding Art Gallery	27
4.4 Minimum Guards	28

4.5 Minimizing and Maximizing Visibility.....	31
4.6 Coverings and Partitioning of Polygons.....	32

Chapter 5: Algorithms

5.1 Problem 1: Given a resource point q , can q see T ?	34
• Specification.....	34
• Notation	35
• Algorithm 5.1	37
• Theorem 5.1	37
5.2 Problem 2: Given a source point q , compute the visible region of T from q	41
• Specification.....	41
• Algorithm 5.2	41
• Theorem 5.2	42
5.3 Problem 3: Does there exist a point q such that q sees T ?	43
• Introduction	43
• Specification.....	44
• Definitions and Notations.....	45
• An algorithm for computing the complete visibility polygon from a polygon inside a polygon with holes.....	46
• Eliminate holes from convex hull of T	49
• Eliminate holes from P	54
• Put it all together.....	59
• Algorithm 5.3	59
• Theorem 5.3	60
5.4 Problem 4: If T is not visible by a single point, then, compute the minimal set of points that sees T	62
• Specification.....	62
• Theorem 5.4.1	70

Chapter 6: Summary and Open Problems

6.1 Summary	71
6.2 Open Questions	73

References	75
------------------	----

Table of Figures

Figure 1.1 Computerized Tomographic scan of the brain.....	3
Figure 1.2 MRI image of the brain.....	3
Figure 1.3 Polygons.....	5
Figure 2.1 Visibility Polygons	12
Figure 3.1 Notations for the configuration.....	16
Figure 3.2 Determining whether p_i should be a vertex of $CV(P,T)$	18
Figure 3.3 Find the visible vertices of P from T	20
Figure 3.4 Illustration for Step 5 II	22
Figure 3.5 Illustration for Step 6 III.....	23
Figure 3.6 An example of $CV(P,T)$ when CT intersects boundary of P	24
Figure 5.1.1 An example of configuration for problem 1	34
Figure 5.1.2 The merger of two visible chains.....	36
Figure 5.1.3 Illustration of Algorithm 5.1	38
Figure 5.3.1 A polygon P , a polygon T and a set of holes	44
Figure 5.3.2 Elements that are inside list I	45
Figure 5.3.3 Three stages for approaching problem 3.....	47
Figure 5.3.4 An example of the list I and the list O	49
Figure 5.3.5: Notations for R_i , S_i and H_i	50
Figure 5.3.6 An example for Lemma 5.3.2	51
Figure 5.3.7 An example of PV	55
Figure 5.3.8 An example for Lemma 5.3.5	55

Figure 5.3. 9 An example for stage 3 of the algorithm	57
Figure 5.4.1 Polynomial reducibility between problem A and B	63
Figure 5.4.2 Restriction on the movement of a radiation source point	65
Figure 5.4.3 Adding a set of holes along the edges of polygon T	66
Figure 5.4.4 Source points are not on the vertices of T	67
Figure 5.4.5 Source points on vertices of T	68
Figure 5.4.6 One radiation source point sees more than one place	69
Figure 5.4.7 Moving the radiation sources to be on the vertices	69

Chapter 1 Introduction

1.1 Motivation

The purpose of this thesis is to identify a way to compute the visibility polygon of a polygon P with holes from a polygon T . The motivation behind it lies in the radiation therapy treatment planning. Radiation therapy is used as one of the treatments for cancer patients whose malignant tumors can not be surgically removed. The aim of radiation therapy is to maximize the high energy x-rays to the cancer cells and to destroy their reproductive capabilities while minimizing the risk of doing damage to the surrounding tissues and healthy organs [40]. For example, in radiation treatment for head and neck cancer patients, care must be taken to minimize the damage to the spinal cord, which could lead to pain, paralysis and/or death. Prior to the radiation therapy, Computerized Tomography (CT) and/or Magnetic Resonance Imaging (MRI) scans are taken to determine the tumor's volume and its surrounding organs. CT uses x-rays to generate a series of cross-sectional images of the body known as tomographic images. Tomographic images are generated from the recorded x-ray projections of a cross section of the body. A 3-dimensional image can be generated from an analysis of the x-ray images taken by the rotating x-ray source (see Figure 1.1). MRI uses strong magnetic fields and radio frequency radiation to form a three dimensional image of a portion of the body. A magnetic field is passed through the body and a rotating scanner is used to measure the

magnetic moments of nuclei (see Figure 1.2). After the tumors are scanned, radiation oncologists will exam the scans, and outline the external contour of the tumor and of other critical structures (if visible within the treatment field). Once the radiation target area has been identified and appropriate margin width is chosen to ensure that surrounding microscopic tissue is included, the spatial information is then transferred to the computer. The computer assists the radiation oncologist to determine the best location for the x-ray source, the dose that shall be used in the target area, and the kind of radiation treatment technique that shall be used. This process is called "treatment planning"[53]. Once a series of possible treatment plans are available, the radiation oncologist may further refine the treatment plans by using digitally reconstructed radiographs (DRR) which are generated from the CT or MRI scans for each of the specified treatment fields. Since most malignant tumors are irregularly shaped or are near organs or other critical structures, the determination of the proper treatment plan is very time consuming. It requires the processing of a great deal of spatial information and to perform a lot of calculations [36]. If it were all done manually, this would take an unacceptably long time for cancer patients to wait for the radiation treatment to begin. Currently, according to Statistics Canada, the average waiting time for a breast cancer patient to begin radiation therapy is 14 weeks. Also, during the radiation therapy period, the radiation region is changing all the time. The measurements need to be up to date for treatment accuracy. Clinically, computers have been used, but because of the complexity of the treatment planning process, more powerful computer systems with more sophisticated algorithms are needed. In fact, if we take the patient's body to be a polygon P , the cancer region to be a polygon T inside P , and the set of normal organs / critical

areas to be a set of obstacles inside P , then the radiation treatment planning problems can be modeled as visibility problems. The goal is to find the visibility region of P from the polygon T in which T is wholly visible in the presence of a set of obstacle polygons.

Our work mainly focuses on finding algorithms for the following questions for a given configuration: Does there exist a single x-ray source point that can cover the whole cancer region without passing through any critical areas? If not, then how much does it cover? Does there exist a source point that covers most of the target area? If not, then how many x-ray source points are required to completely cover the target area without damage to the normal organs? By finding algorithms for the above questions, we believe they will yield optimal solutions for x-ray source location determination.

Figure 1.1 CT scan of the brain

Figure 1.2 MRI image of the brain

1.2 Definitions

Before we go into further details, we will give the terminology and definitions that are necessary in this thesis.

A *polygon* P is defined as a closed finite connected region of the plane bounded by a finite set of n vertices $v_1, v_2, \dots, v_n$ and n edges $v_1 v_2, v_2 v_3, \dots, v_n v_{n-1}, v_n v_1$ such that no pair of non-consecutive edges share a point [49]. The collection of vertices and edges will be referred to as the boundary of P .

A polygon P is *simple* if no pair of nonconsecutive edges of P shares a point. A *simple polygon* partitions the plane into two disjoint regions, the interior (bounded) and the exterior (unbounded) that are separated by the polygon [50].

A *polygon with holes* is a polygon P enclosing several other polygons $H_1, \dots, H_h$, the holes. None of the boundaries of $P, H_1, \dots, H_h$ may intersect, and each of the holes is empty [49].

A *polygonal chain* $ch(u, v)$ is the part of the boundary of a polygon encountered on a traversal of the boundary of P starting at vertex u and ending at v in clockwise order.

A domain D is *convex* if, for any two points q_1 and q_2 in D , the segment $(q_1 q_2)$ is entirely contained in D . A simple polygon P is *convex* if its interior is convex.

Convex Hull The convex hull of a set of points S is the boundary of the smallest convex domain containing S .

Figure 1.3 Polygons

A non-simple polygon

A simple polygon

The convex set of polygon

The convex hull

1.3 Scope of the Thesis

Chapter 2 provides a literature summary about research into the visibility problems. Chapter 3 describes Ghosh's algorithm for computing the visibility polygon from a convex set in detail, which we make use of to solve problem 3 in Chapter 5. Chapter 4 summarizes research results of an art gallery problem that relates to our studies. Chapter 5 presents results of this thesis, including an algorithm for computing the visibility polygon of a polygon with holes from a polygon, and several other algorithms related to the radiation treatment planning applications. Chapter 6 discusses the problems we have solved in Chapter 5 and presents some open problems.

Chapter 2 Visibility Polygon Research

2.1 Introduction

Determining the visible region of a geometric object from a given source under various constraints is a well-studied problem in computational geometry. According to the type of visible source, the visibility polygon problem can be categorized in one of three groups: visibility from a point, visibility from a line segment, or visibility from a polygon.

Given a point set P and two points s and t in P , t is said to be *visible* from s or vice versa if and only if the line segment (s,t) is completely contained in P .

Given a polygon P and a point q in P , the complete visibility polygon $CV(P, q)$ of P from q is the locus of all points of p in P such that p and q are visible. A useful way to understand this is to take the point q as a light source inside a polygonal room P , then $CV(P, q)$ is the illuminated region.

Similarly, the complete visibility polygon $CV(P, xy)$ of polygon P from a line segment (x,y) is the locus of all points of p in P such that p sees every point of the line segment (x,y) . If p sees some points of (x,y) , then, (x,y) is said to be *weakly visible from p* ; the locus of all points of p is called the *weak visibility polygon* of P from line segment (x,y) , denoted as $V(P, xy)$. If we take the edge as a fluorescent light placed on the given edge inside a polygonal room P , then $V(P, xy)$ is the illuminated region.

A simple polygon P is said to be *completely visible from a set of points $T \subset P$* if for every point u in P and every point v in T , u and v are visible (i.e. the line segment (u,v) is contained in P), denoted as $CV(P, T)$. There is no restriction that $CV(P, T)$ contains T . In contrast, P is said to be *weakly visible from a set of points $T \subset P$* if for each point u in P , there exists a point v in T (depending on u) such that u and v are visible, denoted as $WV(P, T)$.

Depending on the location of the visibility source and the definition of the visibility, visibility problems can be further categorized. In this section, we will state the algorithms following the categories below:

1. Visibility region from a point

- Inside a simple polygon
- Inside a polygon with holes
- Outside a polygon

2. Visibility region from a line segment

- Complete/Weak visibility region inside a simple polygon
- Complete/Weak visibility region inside a polygon with holes

3. Visibility region from a polygon

- Inside a simple polygon
- Inside a polygon with holes

Figure 2.1 Visibility Polygons

A complete visibility polygon of P from a point x

$$CV(P, x)$$

A weakly visibility polygon of P from an edge (ab)

$$V(P, ab)$$

A complete visibility polygon of P from an edge (ab)

$$CV(P, ab)$$

2.2 Visibility Region from a Point

2.2.1 Visibility region from a point inside a simple polygon

El Gindy and Avis [21], and Lee [44] have proposed an optimal $O(n)$ time algorithm for computing the visibility polygon from a point inside a simple polygon. Joe [37] discovered that both algorithms may fail if the polygon has a sufficient large winding number. Later, Joe and Simpson [38] presented a correction to Lee's algorithm, and the running time was successfully maintained in linear time.

2.2.2 Visibility region from a point inside a polygon with holes

Lee [44], Suri and O'Rourke [57] gave an $O(n \log n)$ time algorithm for finding the visibility region from a point inside a polygon with holes by using the plane sweep technique, while Arkin and Mitchell [2] used the divide-and-conquer technique to reach the $O(n \log n)$ time. Arkin and Mitchell [2], Asano, Guibas, Hershberger and Imal [3] compute the visible region of the boundary of each hole in linear time and then merged all visible regions. The algorithm takes $O(n \log h)$ time where h is the number of holes inside the polygon. If the holes were either convex or star shaped, Arkin and Mitchell, and Asano used the $\Omega(n + h \log h)$ lower bound sorting technique for computing a visibility polygon provided by O'Rourke and Suri [57], and achieved an $O(n + h \log h)$ time algorithm.

An $O(n + h \log h)$ time algorithm was founded by Heffernan and Mitchell [32] regardless of the shape of the holes. They first justified the problem by giving a reduction of the visibility polygon problem to the visibility profile problem, and then made use of a lid query technique and a dynamic chain insertion algorithm to achieve an optimal algorithm. The visibility profile of a collection of polygons (VP) can be described as a function $y = f(x)$ defined over the domain R , where $f(x)$ is the minimum y coordinate of a point of the polygons with x coordinate x ($f(x) = \infty$ if no such point exists). A lid query is based on planar point location. Basically, given a chain P_i , and a point in the plane p , a lid query of p on P_i asks where p is with respect to P_i . The result of such a query will give users important information about $VP(\{P_i, c\})$ for any chain c containing p such that $c \cap P_i = \emptyset$. The lid query helps to determine quickly if and where a chain c containing a point p lies below the current profile $VP(S)$. It is used in Heffernan and Mitchell's algorithm to merge the set of profiles of polygons (holes) with respect to a point x . Although we made use of the algorithm to achieve an optimal running time to compute a visibility polygon of a polygon P with holes from a polygon T , where T is inside of P , the Heffernan and Mitchell's algorithm was very complex, and the description of the algorithm is beyond the scope of this paper.

2.2.3 Visibility region from a point outside a simple polygon

Lee [44] has extended his linear time algorithm for computing the visibility polygon from a point inside a simple polygon to the case where the given point is outside of the polygon. The running time of the algorithm is also linear. (Refer to Joe and Simpson's paper [38] for corrections.)

2.3 Visibility Region from a Line Segment

2.3.1 Complete/Weak visibility region from a line segment inside a simple polygon

Once we have a linear time algorithm to compute the visibility regions from a given point, we would like to know if we can do the same if the given source is a line segment. If we are looking for a complete visibility polygon from a line segment, then the answer is yes, but if we are looking for a weak visibility polygon, the answer is not that straightforward. Avis and Toussaint [5] proved that the complete visibility polygon of a line segment pq in a polygon is the intersection of $V(P,p)$ and $V(P,q)$. However, if the polygon has a hole, then it is a visibility polygon of q in $V(P,p)$. Therefore, by using Lee's algorithm, we can find $V(P,p)$ of p in P in linear time, compute the $V(P,q)$ of q inside $V(P,p)$, and have $V(P,pq)$ in $O(n)$ time. In the case of computing a weak visibility polygon inside a simple polygon, El Gindy [22], Lee and Lin [45], and Chazelle and Guibas [13] proposed $O(n \log n)$ time algorithms. Guibas, Hershberger, Leven, Sharir and Tarian [29] presented an $O(n \log \log n)$ time algorithm for the same problem but this requires a triangulation of the polygon first. There are several fast deterministic and randomized algorithms giving bounds of $O(n \log \log n)$ [41], [58] or $O(n \log^* n)$ [17], [54] for triangulation of a simple polygon. Chazelle gave a linear-time algorithm, thereby improving Guibas et al.'s algorithm running time to linear as well. In conclusion, a linear time algorithm for computing a visibility polygon from a line segment inside a simple polygon exists. The next question is what about the line segment inside a polygon with holes.

2.3.2 Complete/Weak visibility region from a line segment inside a polygon with holes

Suri and O'Rourke [57] presented an algorithm for computing the weak visibility polygon from a line segment inside a polygon with holes that runs in $O(n')$ time. They also showed that the algorithm is worst-case optimal: there are polygons with holes that contain a line segment whose visibility polygon has $\Omega(n')$ vertices.

2.4 Visibility Region from a Polygon

2.4.1 Visibility region from a polygon inside a simple polygon

All of the above algorithms compute the visibility polygon either from a point or from a line segment. It is natural to generalize the notion of visibility from a point or a line segment to that of visibility from a polygon. Ghosh [28] studied the problem and presented an $O(n + k)$ algorithm. It can be used for both the complete and weak visibility polygon computations, in a configuration of a polygon T with total of k vertices inside a polygon P with total of n vertices. Ghosh assumed that the complete visibility polygon must contain polygon T . This is not necessarily so as we will discuss in Chapter 3. Therefore, we present an algorithm which only restricts the complete visibility polygon inside P but does not necessary contain T . Since our algorithm applies Ghosh's algorithm if the condition is met, we therefore will give a detail of the algorithm in Chapter 3.

2.4.2 Visibility region from a polygon inside a polygon with holes

This is an open problem. There is no algorithm yet has been published for computing the visibility polygon of a polygon T inside another polygon P with holes. We will present our research result in Chapter 5.

Chapter 3 Algorithm for Computing the Visibility Polygon from a Polygon inside a Simple Polygon

This chapter describes Ghosh's algorithm [28] for computing the visibility polygon from a polygon inside a simple polygon. We will give a detailed description of the algorithm's objective, its operation, and its impact to our algorithms in this chapter. We will make use of it in our algorithm described in Chapter 5.

3.1 Specification

Given:

- A simple polygon T with k vertices
- A simple polygon P with n vertices
- The polygon T is inside the polygon P

Task:

- Compute the **complete visibility polygon** of P from T .

Output:

- A list of vertices which is the boundary of the complete visibility polygon of P from T

3.2 Definitions and Notations

Before we describe the algorithm, we will give some definitions and notations as required for the description of the algorithm.

- Let $t_1, t_2, \dots, t_k$ be the sequence of vertices listed clockwise around the boundary of T and let $p_1, p_2, \dots, p_n$ be the sequence of vertices listed clockwise around the boundary of P .
- Let $CV(P, T)$ be the complete visibility polygon of P from the polygon T . For every point $u \in CV(P, T)$, and every point v of T , u and v are visible.
- Let $CV(P, t_i)$ be the complete visibility polygon of P from point t_i . For every point $u \in CV(P, t_i)$, u and t_i are visible.
- $Ray[a, b]$ is defined as the half-line originated at a point a to another point b .
- Each vertex p_i has one left and one right tangent line with respect to the polygon T . Segment $p_i t_j$ is left tangent of p_i if all points of T lie to the right of the $Ray[p_i, t_j]$, and segment $p_i t_j$ is right tangent of p_i if all points of T lie to the left of the $Ray[p_i, t_j]$.

Figure 3.1 Notations for the configuration

3.3 Introduction

Let S be the set of vertices of the convex hull polygon T inside P . Ghosh showed that in order to find the complete visibility polygon of P from T , $CV(P, T)$, it is enough to find the complete visibility polygon of P from S , $CV(P, S)$. Assuming that the boundary of the region T is given as a clockwise sequence of vertices $t_1, t_2, \dots, t_k$, a brute force approach to compute $CV(P, T)$ is as follows: Find the complete visibility polygon of P , say $CV(P, t_1)$, from a vertex t_1 of T , using Lee's linear time algorithm as discussed in Chapter 2, then compute the complete visibility polygon of P from the vertex t_2 in $CV(P, t_1)$, which means $CV(P, t_2) = (CV(P, t_1), t_2)$, and continue in this manner until all vertices of T have been visited, such that $CV(P, T) = (CV(\dots (CV(P, t_1), t_2), \dots), t_k)$. Since computing each visibility polygon from a vertex of T takes $O(n)$ time by using Lee's algorithm [44], there are k vertices of T repeatedly applying Lee's algorithm. Thus, the total running time for computing the $CV(P, T)$ is $O(kn)$ time.

Ghosh presented an improved solution which takes only $O(n + k)$ time in the worst case, where n is the number of the vertices of P , and k is the number of the vertices of T . The main idea behind his algorithm is that each pair of tangent lines of vertex p_i of P with respect to polygon T , namely $Ray[p_i t_i]$ and $Ray[p_i t_j]$, determines whether p_i should be a vertex of $CV(P, T)$ depending on if the tangent line intersects the boundary of P or not.

An intersection among the tangent lines of each vertex p_i with the boundary of P will be one of the following cases

- The tangent lines of p_i do not intersect the boundary of P .

This indicates that p_i is visible. Vertex p_i will be part of the boundary of $CV(P, T)$.

- The tangent lines of p_i intersect the boundary of P more than once.

This indicates that p_i is not visible, therefore, vertex p_i should not be in the $CV(P, T)$.

Figure 3.2 Determining whether p_i should be a vertex of $CV(P, T)$

In Figure 3.2, vertex p_2 's two tangent lines $Ray[p_2, t_1)$ and $Ray[p_2, t_3)$ do not intersect the boundary of P , so p_2 is visible from the polygon T . Therefore, p_2 will be a vertex of $CV(P, T)$. On the other hand, p_4 's left tangent line $Ray[p_4, t_4)$ intersects the edge p_5p_6 of P , then vertex p_4 is not visible from the polygon T . Therefore, vertex p_4 will not be a vertex of $CV(P, T)$.

Instead of visiting each vertex of T one by one to compute $CV(\dots CV(CV(P, t_1), t_2), \dots t_k)$, Ghosh's algorithm does the follows. Firstly, all vertices of P are stored in a list. Then, the algorithm examines the right tangent lines of all vertices in the list. If the tangent line intersects the boundary of $CV(P, t_1)$, that vertex will be removed. At the end of this step, each p_i whose right tangent line does not intersect the $CV(P, t_1)$ will remain in the list,

denoted as VP list. Secondly, the algorithm examines all left tangent lines of the vertices that are stored in the VP list. Again, any vertices in the VP list whose left tangent line intersects with the boundary of $CV(P, t_i)$ will be removed. Once this step is completed, all vertices that remain in the VP list are the vertices of the $CV(P, T)$. In this way, all vertices of P will be examined at most twice, one is clockwise direction for the right tangent line, and the other one is counterclockwise direction for the left tangent line. Therefore, the running time of the algorithm is reduced significantly.

The process to find out if a vertex of P is contained in the $CV(P, T)$ appears at first to be very complicated. It is simple as outlined in Ghosh's paper. For every vertex of P , the clockwise scan checks if p_i lies to the left of $Ray[t_j, p_{i-1}]$. If so, then, somewhere along the boundary after p_i , the boundary of P has to come back to some vertex z and intersects the $Ray[t_j, p_{i-1}]$, because the boundary of P is a closed curve. Therefore, $Ray[t_j, p_{i-1}]$ intersects the boundary of P at point p_{i-1} and z , which means p_i is blocked by p_{i-1} . Thus, vertex p_i is not visible from T (see Figure 3.3), which means p_i will not be in $CV(P, T)$. Similar operation is performed in the counterclockwise direction for vertices remaining in the VP list.

Figure 3.3 Find visible vertices of P from T

In Figure 3.3, vertex p_8 lies to the left of the Ray $[t_5, p_7)$, the Ray $[t_5, p_7)$ intersects the boundary of P at a point z . It means that p_8 is blocked by p_7 , therefore, vertex p_8 is not visible by t_5 , and it will not be contained in the VP list. This implies that p_8 will not be a vertex of $CV(P, T)$.

3.4 Description

The following is a step-by-step description of the algorithm.

1. Given a simple polygon T inside a simple polygon P . If T is non-convex, find the convex hull of T . If the boundary of the convex hull of T intersects the boundary of P , then stop and return empty. Otherwise, go to the next step.
2. Initialize a double linked list, VP , with vertices of the boundary of P .
3. Compute $CV(P, t_i)$ using Lee's algorithm.
4. Compute p_c and p_{cc} .

5. Now, travel along the boundaries of P and of T in a clockwise order, starting from p_{c+1} of the boundary of P and t_1 of boundary of T , do the following:
 - I. If $p_i = p_c$, we have visited all vertices of P ones, stop.
 - II. While p_i lies to the right of $\text{Ray}[t_j, t_{j+1})$, increment j till p_i lies on the left of the $\text{Ray}[t_j, t_{j+1})$.
 - III. While p_i lies to the left of $\text{Ray}[t_j, p_{i-1})$, assign i to s , increment s till p_s, p_{s+1} intersects $\text{Ray}[t_j, p_{i-1})$ at some point z , then replace $(p_i, p_{i+1} \dots p_s)$ with z in VP . Point z becomes p_i .
 - IV. If p_i lies to the right of or on the $\text{Ray}[t_j, p_{i-1})$, then, p_i is visible, in this case, no additional action is taken other than incrementing i , moving on to next vertex of P .
6. Now, travel along the boundary of T in the counter clockwise order starting from t_1 , and visit from last to first vertices of P stored in VP . Do the following:
 - I. If $p_i = p_{cc}$, then stop.
 - II. While p_i lies to the left of $\text{Ray}[t_j, t_{j-1})$, decreasing j till p_i lies on the right of the $\text{Ray}[t_j, t_{j-1})$. Segment p_i, t_j becomes the left tangent of p_i .
 - III. While p_i lies to the right of $\text{Ray}[t_j, p_{i+1})$, then assign i to s , decreasing s till p_s, p_{s-1} intersects $\text{Ray}[t_j, p_{i+1})$ at some point z . Then replace $(p_i, p_{i-1} \dots p_s)$ with z in VP . Point z becomes p_i (see Figure 5.4).
 - IV. If p_i lies to the left of or on the $\text{Ray}[t_j, p_{i+1})$, then, p_i is visible, decreasing i , moving on to next vertex of P .
7. After Step 5 and 6, VP contains the boundary of the complete visibility polygon of P from T . Return VP .

Vertex p_c is the closest point to t_l among the intersection points of $\text{Ray}[t_k, t_l]$, and p_{cc} is the closest point to t_l among the intersection points of $\text{Ray}[t_2, t_l]$. Step 5 and 6 will only work when $CV(P, t_l)$ exists. For the simplicity of the algorithm, assume that $CV(P, t_l)$ in Step 3 exists. Step 5 II (resp., Step 6 II) is searching for the right (resp., left) tangent line of p_i . When all points of T from t_l to t_j all lie on the right (resp., left) of $\text{Ray}[t_j, t_{j+1}]$, then $p_i t_j$ is the right (resp., left) tangent line of p_i . We will explain it using Figure 3.4. Step 5 III and Step 6 III removes all vertices that are contained in the VP list but are not visible by t_j (see Figure 3.5)

Figure 3.4 Illustration for Step 5 II.

Let $p_i = p_4$, $\text{Ray}[t_j, t_{j+1}]$ is the $\text{Ray}[t_1, t_2]$. Since p_4 lies to the right of the $\text{Ray}[t_1, t_2]$, therefore, increase j from 1 to 2, now, vertex p_4 lies to the left of the $\text{Ray}[t_2, t_3]$, then, segment $p_4 t_2$ is the right tangent line of p_4 .

Figure 3.5 Illustration for Step 6 III

Let $p_i = p_6$ and $t_j = t_5$. Since p_6 lies to the right of the $\text{Ray}[t_5, p_7]$, assign 6 to s , decreasing s till $\text{Ray}[t_5, p_7]$ intersects edge p_5p_4 at a point z , replacing p_6, p_5 with the point z in the VP list.

3.5 Running-time Analysis

Computing the $CV(P, t_1)$ in Step 3 takes $O(n)$ time by using Lee's computing the visibility polygon from a point algorithm. Steps 5 and 6 scan vertices of P and vertices of T at most twice. Each vertex takes $O(1)$ time. Therefore, the total running time of algorithm is $O(n + k)$ time.

3.6 Discussion

The algorithm is simple and easy to implement. Ghosh claimed that if the boundary of the convex hull of the inside polygon T intersects the boundary of the polygon P , then the complete visibility polygon of P from T is empty. This is true only if the complete visibility polygon has to contain the polygon T , or the algorithm is being used for computing a minimum link path between any given two points inside the polygon P in linear time. This is not true when these conditions are removed, and the definition of $CV(P, T)$ still applies (see Figure 3.6).

Figure 3.6 An example of $CV(P, T)$ when CT intersects boundary of P

Given a polygon T and a polygon P The convex hull of T intersects the boundary of P

Any point in the shaded area sees the entire polygon T

Chapter 4 The Art Gallery Problem

4.1 Introduction

In 1973, V. Klee raised a question: How many guards are required to sufficiently guard any polygon with n vertices? Soon after, V. Chvatal [16] established what has become known as Chvatal's Art Gallery Theorem: $\lfloor n/3 \rfloor$ guards are always sufficient and occasionally necessary to cover a simple polygon with n vertices. Since then, the Art Gallery Theorem has been extended and a lot of algorithms have been developed. In 1987, J. O'Rourke [49] published *Art Gallery Theorems and Algorithms*, the first book dedicated solely to the study of the guarding problems of polygons on the plane. In 1992, T. Shermer [56] published a thorough survey paper on the Art Gallery problem. Since our project focuses on the visibility of a polygon, and some of the problems relate to the Art Gallery Theorem, we therefore will cover those related research results published to date. Visibility graphs studied in O'Rourke's book and Shermer's survey will not be covered in this chapter.

4.2 Basic Concepts

A set H of points **guards** a polygon P if every point u of P is visible from a point p in H .

A way to understand it is to station a guard at each element of H , if P is totally visible, then, P is guarded.

Based on the problem requirements, guards can be positioned differently to cover a polygon P . They may be located anywhere within P , or patrol along diagonals or edges of P .

Point Guards: Guards can be located anywhere inside of a polygon P . The set H of guard points is a subset of the polygon P .

Vertex Guards: Guards can only be placed at the vertices of the polygon P .

Edge Guards: Guards can move along the edges of the polygon P to guard the polygon P . A point q can be considered guarded if it is visible from some points in the path of a guard.

Mobile Guards: Guards are allowed to move along closed line segments totally contained in a polygon P .

4.3 Guarding Art Gallery

S. Fisk [24] proved Chvatal's Art Gallery Theorem by triangulating the polygon P first, then 3-coloring the resulting graph, and finally stationing a guard at all the vertices of the smallest chromatic class. A k -coloring of a graph is an assignment of colors to the nodes, one color per node, using no more than k colors, such that no two adjacent nodes are assigned the same color. This has shown that the triangulation of a polygon plays a central role in the study of the Art Gallery problem. A *diagonal* of a polygon is a line segment whose endpoints are a pair of visible vertices. A *triangulation of a polygon* is a partition of the polygon into triangles by diagonals. One can also divide a polygon into triangles without requiring that the partitioning segments be diagonals, i.e. by partitioning along any set of appropriately chosen line segments inside the polygon. In such a triangulation, there may be endpoints of line segments that are not vertices of the polygon (such endpoints are called Steiner points), so this types of triangulation is called a triangulation with Steiner points [52]. A lot of algorithms for solving the Art Gallery Theorem use the triangulation technique. In 1978, Garey, Johnson, Preparata and Tarjan [25] presented the first $O(n \ln n)$ time triangulation algorithm. Avis and Toussaint [6] gave an $O(n \ln n)$ time algorithm to solve the art gallery problem by using their triangulation algorithm. Tarjan and Van Wyk [58] improved the triangulation algorithm to $O(n \log \log n)$ time in 1988. Chazelle [14] developed an optimal linear-time triangulation algorithm in 1991.

4.4 Minimum Guards

As stated in the previous section, the art gallery theorem can be converted into an algorithm that covers a polygon of n vertices with $\lfloor n/3 \rfloor$ guards in a linear time. Although $\lfloor n/3 \rfloor$ guards are necessary in some cases, often there are far more guards than are needed to cover a particular polygon. It is natural, then, to seek a placement of a minimal number of guards that covers a given polygon. This is a much harder problem. Lee and Lin [43] in 1979 have proved that the minimum vertex guard problem for polygons is NP-hard. In the same paper, they have also shown that the minimum edge and point guard problems are also NP-hard.

If an algorithm can solve a problem in a polynomial time, the algorithm is called polynomial-time algorithm, and the problem is tractable. However, not all problems can be solved, no matter how much time is provided. Also, there are problems that can be solved, but not in time $O(n^k)$ for any constant k . Problems that can be solved in non-deterministic polynomial (NP) time are generally intractable. The hardest problems in NP, whose statuses are unknown, are called the "NP-complete" problems.

If a problem A can be transformed in polynomial time to an instance of problem B such that the solution to B yields a solution to A , then A is said to be polynomial reducible to B . It shows that problem B is at least as hard as problem A , to within a polynomial-time factor.

A problem B is *NP-complete* if

1. Problem $B \in NP$ and
2. Problem A is polynomial time reducible to problem B , for every $A \in NP$.

If problem B satisfies property 2, but not necessarily property 1, we say that problem B is *NP-hard*.

Up to now, no polynomial-time algorithm has yet been discovered for a NP-complete problem, nor has anyone yet been able to prove a super-polynomial time lower bound for any of them. If a problem is NP-complete, rather than searching for a fast algorithm that solves the problem exactly, we are better off finding an approximation algorithm.

O'Rourke [48] proved that any polygon with n vertices and h holes can always be guarded with $\lfloor (n+2h)/3 \rfloor$ vertex guards. Bjorling-Sachs and Souvaine [8] and Hoffmann, Kaufman, and Kriegel [34] independently proved that $\lceil (n+h)/3 \rceil$ point guards are always sufficient and occasionally necessary to guard any polygon with n vertices and h holes. Bjorling-Sachs and Souvaine [9] gave an $O(n^2)$ algorithm to position those $\lceil (n+h)/3 \rceil$ point guards. The main idea in Bjorling-Sachs and Sourvaine's paper is to connect each hole of the polygon to the exterior with a quadrilateral "channel" and then to triangulate the hole-free version of the polygon. One vertex is introduced for each channel, and there is a triangle T in the remaining polygon such that any point in it sees all of the channel. The triangulation of the hole-free version of the polygon is not arbitrary. The triangle T

is forced to be included with the result that a guard placement based on 3 coloring in the hole-free polygon will automatically cover the channel.

An orthogonal polygon is one whose edges are all aligned with a pair of orthogonal coordinate axes, which we take to be horizontal and vertical without loss of generality [49]. Kahn, Klawe and Kleitman [39] have proved that any orthogonal polygon with n vertices can always be covered with $\lfloor n / 4 \rfloor$ vertex guards. $\lfloor n / 4 \rfloor$ are sometimes necessary. The technique for the proof is similar to Fisk's 3-coloring approach. They first partition the orthogonal polygon into convex quadrilaterals, then add internal diagonals to each of those quadrilaterals, finally, 4-coloring the graph. Sack [51] has obtained an algorithm for partitioning a rectilinear polygon into convex quadrilaterals in $O(n)$ time. Edelsbrunner, O'Rourke and Welzl [20] gave a linear time algorithm to guard an orthogonal polygons using $\lfloor n / 4 \rfloor$ guards by partitioning the orthogonal polygon into L-shaped polygons.

By allowing the guards to move along diagonals joining vertices of P , O'Rourke was able to prove $\lfloor n/4 \rfloor$ mobile guards are always sufficient and occasionally necessary to guard any polygon with n vertices [48]. Shermer [55] has proved that $\lfloor 3n/10 \rfloor$ edge guards are always sufficient and occasionally necessary to guard any triangulation graph with n vertices, with the exception of three graphs.

Using the consecutive edges or vertex sets to cover the polygon P has also been studied. Avis and Toussaint [5] first gave an optimal linear time algorithm to determine the visibility polygon from an edge e of P (for other related algorithms, refer to Chapter 2).

Abellanas, Garcia-Lopez and Hurtado [1] showed that for n even (resp. n odd) $n - 3$ consecutive vertex guards (resp. $n - 4$) are always sufficient and occasionally necessary to guard any polygon with n vertices, $n \geq 4$. They also showed that $n - 5$ consecutive edge guards are always sufficient and occasionally necessary to guard any polygon P with n vertices, when $n \geq 5$. For $n \leq 5$, one edge guard suffices. They also presented an $O((n^2) \log n)$ time algorithm to compute the minimum number of consecutive vertex and edge guard sets. Chen, Estivill-Castro and Urrutia [15] proved that the shortest guarding boundary chain and the smallest guarding set of consecutive vertices of a polygon with n vertices can be found in $O(n \ln n)$ time. The problem of finding the minimum number of consecutive vertex and edge guard sets for a polygon with holes is open. No result has been published yet.

4.5 Minimizing and Maximizing Visibility

When there is only one guard available, the best position is where the guard has the largest visibility. On the other hand, if we want to hide a treasure inside a polygon, the best place is where the guard has only the minimal visibility. The problem of finding the location for a guard inside a polygon in such a way that the area covered by the guard is maximized or minimized has also been studied. Ntafos and Tsoukalas [47] proved that finding a vertex guard of a polygon P with n vertices that maximized visibility can be done in $O(n^2)$ time. Finding the point along the boundary of P that maximizes the area of visibility can be found in $O(n^3 L)$ where L is the time it takes to numerically solve an equation of degree $O(n)$. Gewali [26] showed that finding the point on the boundary of

an orthogonal polygon that minimizes visibility can be done in $O(n^3)$ time and $O(n^3 \ln n)$ for orthogonal polygons with holes.

When the guard's position is not restricted to the vertex or to the boundary of polygon P , then the problem of finding the optimal point location of a guard that maximizes the visibility area is open. It is not known if an efficient algorithm to solve it exists.

4.6 Coverings and Partitioning of Polygons

A polygon P is called *star shaped* if there is a point $q \in P$ such that all the points of P are visible from q . The *kernel* of a polygon P is the set of all points that see all of P . It follows by definition that star-shaped polygons are those that can be guarded with one guard. Lee and Preparata [42] has showed that the kernel of a polygon can be found in linear time. Culberson and Reckhow [19] showed that finding the smallest number of convex polygons needed to cover a simple polygon is NP-hard. They also showed that the related problems of covering only the edges or vertices of a polygon are also NP-hard. Hecker and Herwing [31] showed that Diagonal Guarding, Triangulation Triangle Guarding and Mobile Guarding are all NP-hard. Chazelle and Dobkin [11] proved that the problem of finding a minimum convex partitioning of a polygon can be done in polynomial time. They proved that finding a minimum convex partition of a simple polygon P with n vertices into convex pieces can be done in $O(n + N^3)$ time, where N is the number of reflex vertices of P . Lingas [46] has showed that this problem becomes NP-hard for polygons with holes.

Chapter 5: Algorithms

Let polygon T represent a tumor with k vertices, and let a set h of holes represent a set of healthy organs. Let P be a polygon, which contains T and h , and represent the body of the patient. A point q represents a radiation source point, which is located outside the tumor polygon T . We would like to find answers for the following problems:

Problem 1: Given a source point q , can q see T ?

Problem 2: Given a source point q , compute the visible region of T from q

Problem 3: Does there exist a point q such that q sees T ?

Problem 4: If T is not visible by a single point, then, compute the minimal set of points that sees T .

5.1 Problem 1: Given a source point q , determine if q sees the entire polygon T .

Specification

Given:

- A polygon T with k vertices
- A polygon P contains a set h of holes with n vertices in total
- A point q outside the polygon T

Compute:

- Determine if the complete visibility polygon from a point q contains the entire polygon T

Output

- True or False

Figure 5.1.1 An example of configuration for problem 1

Notation

- Let $CV(P, q)$ be the complete visibility polygon of polygon P from a point q .
- Let $VC(H, q)$ be the set of visible chains from a point q with respect to the set of holes.
- Let $V(BP, q)$ be the boundary of P that is visible from a point q . When point q is outside of P , assuming point q is at $y = -\infty$, $V(BP, q)$ contains two vertices, l_i and r_i of a minimum and maximum x coordinate, respectively. We define that the counterclockwise chain of $V(BP, q)$ from l_i to r_i is the lower chain of $V(BP, q)$, and the clockwise chain of $V(BP, q)$ from l_i to r_i is the upper chain of $V(BP, q)$. The lower chain of $V(BP, q)$ completely blocks the upper chain of $V(BP, q)$ from the view of the point q , and it also blocks $VC(H, q)$. We therefore define two points as *x-ray visible* if the straight-line segment between them only crosses the lower chain of $V(BP, q)$.
- Let P_i and P_j be two visible chains from point q , where q is at $y = -\infty$ and two chains' x coordinate domains overlay. Without loss of generality, assume that the minimum x coordinates of P_i , $x(l_i)$ is less than the minimum x coordinates of P_j , $x(l_j)$, the merger of these two chains, $VC(i, j)$, has the following characterizations (see Figure 5.1.2):
 1. If the maximum x coordinates of P_i , $x(r_i)$ is less than the maximum x coordinates of P_j , $x(r_j)$, then $VC(i, j)$ consists of a piece from P_i lying left of a piece from P_j .
 2. If $x(r_i) < x(r_j)$, there are two possibilities:
 - If P_i lies completely above P_j , then $VC(i, j)$ is the single piece P_i .
 - If P_j appears once in $VC(i, j)$, then P_i must appear exactly twice, since pieces alternate, and the left and right most pieces are from P_i .

Figure 5.1.2: Merger of two visible chains, P_i and P_j .

(1)

(2)

(3)

Algorithm 5.1 for Problem 1:

1. Compute $VC(H, q)$.
2. Compute $V(BP, q)$.
3. Merging $VC(H, q)$ and $V(BP, q)$, generates $CV(P, q)$.
4. Test if $T \subseteq CV(P, q)$.

Theorem 5.1: Algorithm 5.1 solves problem 1 in $O(n + k + h \log h)$ time.

Proof: According to the definition of $CV(P, q)$, every point in $CV(P, q)$ sees point q . If $T \subseteq CV(P, q)$ then, every point of T and q are visible. Therefore, q sees T completely.

Step 1, we compute a set of visible chains with respect to the set of holes from a point q . This can be done in $O(h)$ time using Lee's [44] linear time algorithm, where h is the number of the vertices of the set of holes.

Step 2, we compute the boundary of P that is visible from the point q . Point q can be inside or outside of polygon P . For both cases, the visible boundary of P can be done by applying Lee's [44] linear time algorithm.

Step 3, we can apply Heffernan and Mitchell's [32] algorithm directly if point q is inside polygon P . If point q is outside of P , we first merge the upper chain of $V(BP, q)$ with

$VC(H, q)$ using Heffernan and Mitchell's [32] algorithm, which runs in $O(h \log h + n)$ time. It results a visible chain from point q . We then merge this visible chain with the lower chain of $V(BP, q)$ in $O(n)$ time (see Figure 5.1.2).

To determine if $T \subseteq CV(P, q)$, we can use Finke and Hinrichs's [23] overlaying simple connected planar subdivision technique. It takes $O(N)$ time, where N is the total number of edges of $CV(P, q)$ and T . It is possible that point q sees all n vertices of polygon P . In addition, when $VC(H, q)$ merges with the boundary of P , each of visible chain's two supporting lines with respect to point q intersects the boundary of P , resulting in additional $2h$ vertices in the boundary of $CV(P, q)$. Therefore, total number of vertices of $CV(P, q)$ is at most $(n + 2h)$. Polygon T has k vertices. Thus, this step takes $O(n + 2h + k)$ time.

The total running time for Algorithm 5.1 is $O(n + h \log h + 2h + k) = O(n + h \log h + k)$ time.

Figure 5.1.3: Illustration of Algorithm 5.1

a) A given configuration with point q is outside of polygon P

- b) Computing the set of visible chains from a point q with respect to the set of holes, $VC(H, q)$.

- c) Compute the visible boundary of P from point q , $V(BP, q)$

d) Merge the $VC(H, q)$ with $V(BP, q)$

i) merge the $VC(H, q)$ with the upper chain of $V(BP, q)$

ii) merge with the lower chain of $V(BP, q)$

e) Test if $T \subseteq CV(P, q)$, in this example, a "No" will be returned.

5.2 Problem 2: Given a source point q , compute the complete visible region of T from q

Specification

Given:

- A polygon T with k vertices
- A set h of holes with n vertices in total
- A point q outside polygon T

Compute:

- The portion of T that can be seen by point q

Output

- The boundary of T that is visible from point q

Algorithm 5.2 for Problem 2:

1. Compute $CV(P, q)$
2. Compute $T \cap CV(P, q)$

Theorem 5.2: Algorithm 5.2 solves problem 2 in $O(n + k + l + h \log h)$ time, where l is the number of intersections of the output.

Proof: The correctness of the algorithm follows Theorem 5.1 thus all that needs to be shown is the boundary on the running time. As we have proven in Theorem 5.1, the first step of the algorithm takes $O(n + h \log h)$ time.

In the second step, we again can apply Finke and Hinrichs' [23] overlaying simple connected planar subdivision technique which requires $O(N + l)$ time. The Finke and Hinrichs' algorithm computes the overlay of two simple connected planar subdivision, subdivision Π_b and subdivision Π_g . The overlay $\Pi_b \cap \Pi_g$ consists of all non-empty sets of intersected edges of the two subdivisions. It runs in $O(N + l)$ linear time and space, where N is the total number of edges of Π_b and Π_g , which is $(n + 2h + k)$ as proven in Theorem 5.1, and l is the number of intersects between the two subdivision edges.

The total running time of Algorithm 5.2 is $O(n + h \log h + k + l)$.

5.3 Problem 3: Does there exist a point q such that q sees T ?

5.3.1 Introduction

The solution to this problem is harder than those of the first two problems. No point is given in advance, we do not know if such point exists until we examine the given configuration. We need to find the complete visibility polygon of P from the polygon T . If such visibility polygon exists and contains points on the boundary of P , then the answer to problem 3 is yes (since the source point q has to be outside the body), else the answer is no. No result has been published for solving this problem as discussed in Chapter 2. In this section, we present our algorithm with the following steps. In Step 1, we compute two lists I and O . List I contains sub-polygons of holes and the boundary of P (if there is any) that are inside the convex hull of polygon T . List O contains all sub-polygons of holes that are outside of the convex hull of polygon T . In Step 2, we compute the visible region of P from T with respect to I , and the visible region of P from T with respect to O . In Step 3, we compute the intersections of two visible regions which are obtained from Step 1 and 2 to obtain the complete visibility polygon of P with respect to T .

5.3.2 Specification

Given

- A polygon P with a set h of holes and n vertices
- A polygon T with k vertices
- Polygon T is inside polygon P

Compute

- Determine the region for all single source points from which the tumor polygon T is entirely visible.

Output

- A complete visibility polygon of P from T

Figure 5.3.1 A polygon P , a polygon T and a set of holes

5.3.3 Definitions and Notations

- Let CT be the convex hull of polygon T .
- Let $CV(P, x)$ be the complete visibility polygon of P from a point x .
- Let $CV(P, T)$ be the complete visibility polygon of P from a polygon T , where T is inside the polygon P .
- Let $C = \{C_1, C_2, \dots, C_m\}$ be a set of chains of the boundary of polygon P , which is contained inside CT .
- Let $H = \{H_1, H_2, \dots, H_j\}$ be a subset of holes that is contained inside CT .
- Let $I = C \cup H$ be a list that contains the set of chains of P and holes that are inside CT . Each element of I contains in a pocket of T defined by convex hull edge (e_u, e_v) .
- Let O be a list that contains the set of polygonal holes that is outside of CT .
- Let $P \setminus O$ be the polygon P without containing elements of O .

Figure 5.3.2: Elements that are inside list $I = \{H_1, H_2, H_3, H_4, C_1\}$

5.3.4 An algorithm for computing the complete visibility polygon from a polygon inside a polygon with holes

Lemma 5.3.1: Given a polygon T inside a polygon P , we claim that $CV(P, T) \subseteq CV(P, x)$, where $x \in T$.

Proof: It suffices to show that every point of $CV(P, T)$ is in $CV(P, x)$, where $x \in T$. Let u be a point in $CV(P, T)$. Since $x \in T$, so u is visible from x , segment ux lies entirely in $CV(P, x)$, so, $u \in CV(P, x)$

Given a polygon P and a polygon T , where T is inside of P , Ghosh [28] has presented an algorithm to compute $CV(P, T)$, a complete visibility polygon of P from T . The algorithm only holds for the case that the inside polygon T is convex, or that the boundary of CT does not intersect the boundary of P . The result of $CV(P, T)$ must contain the entire T . Also, Ghosh's algorithm does not deal with polygon P with holes. In our case, with respect to the position of CT , P and the set of holes, there are five possible cases:

1. Polygon T is convex. CT intersects neither P nor any holes.
2. Polygon T is non-convex
 - a) CT intersects P , but not any of its holes.
 - b) CT intersects P and some of its holes.
 - c) CT does not intersect P , but some of its holes.
 - d) CT does not intersect P or any hole.

Not all of the above cases meet Ghosh's algorithm requirement, therefore we can not apply Ghosh's algorithm directly. Also, the result of our algorithm does not require that $CV(P, T)$ has to contain the polygon T . However, if we can alter the polygon T to be convex and eliminate the holes from the configuration, the problem can be modeled to find the complete visibility polygon of P *without holes* from a *convex* polygon T . The solution for problem 3 will be simplified significantly. We therefore approach the problem in three stages:

1. *Eliminate holes from CT .*
2. *Eliminate holes from the polygon P .*
3. *Compute $CV(P, T)$.*

Figure 5.3.3: Three stages for approaching problem 3

a) A given configuration

b) Step 1: *Eliminate holes from CT*

c) Step 2: *Eliminate holes from P*

d) Step 3: *Computing $CV(P, T)$ of P without holes from a convex T*

5.3.4.1 Eliminate holes from CT

Given a polygon P with a set of holes, and a polygon T inside P , we first split the set of holes into two lists, I and O . Sub-polygons of the holes that are contained inside CT are stored in list I . Sub-polygons of holes that are outside of CT are stored in list O . If the boundary of P intersects the boundary of CT , store the chains of P that are inside of CT in list I . Let $C = \{C_1, C_2, \dots, C_m\}$ be the set of chains of P that is contained inside CT , and let $H = \{H_1, H_2, \dots, H_j\}$ be the subset of holes that is contained inside CT , then list $I = C \cup H$ (see Figure 5.3.4). After the splitting, we have made T convex. Thus CT contains polygon T and elements of I .

Figure 5.3.4 An example of the list I and the list O .

List I contains H_1 and H_2

List O contains H_1, H_2 and H_3

Let H_i be one element of I contained in a pocket of T defined by convex hull edge (V_u, V_l) . Let R be the relative convex hull of T with respect to I . let $R_i = \{V_u, \dots, V_l\}$ be the chain of R between V_u and V_l that separates H_i from T . For each edge e of R_i , let $S_i = \{S_{i1}, S_{i2}, \dots, S_{il}\}$ be the half-planes bounded by the supporting line of e and away from H_i . We define the shag of T with respect to H_i , denoted as $SHAG(T, H_i)$, as the intersections of all such S_i away from H_i . Let $V(T, H_i)$ be the locus of all points of p from which T is visible with respect to H_i .

Figure 5.3.5: Notations for R_i , S_i and H_i

Lemma 5.3.2: $SHAG(T, H_i) = V(T, H_i)$ for $1 \leq i \leq m$, where m is the number of sub-polygonal holes inside CT .

Proof: We show, for a point q , that $q \in SHAG(T, H_i)$ if and only if $q \in V(T, H_i)$.

If $q \in SHAG(T, H_i)$, then all of the chains of R_i are visible from q . Furthermore, if we extend the boundary of the chain R_i to infinity, denoted as R_i' , q still sees this infinite

chain R_i' . Since T is entirely on one side of R_i' and R_i' is completely visible from q . There is no obstacles between the chain R_i and the polygon T . Therefore, T is also completely visible from q .

Figure 5.3.6 An example for Lemma 5.3.2

Conversely, suppose $q \notin SHAG(T, H_i)$, then $q \notin S_i$ for some edge $e = (u, v)$. There are three possible cases:

Case 1: u is a vertex of H_i and v is a vertex of T .

Case 2: Both u and v are vertices of H_i .

Case 3: Both u and v are vertex of T .

Case 1: Let l be the supporting line of e . We claim that l must intersect T in a point a such that a and v are on opposite sides of u on l . If this were not the case, the edge e would not be part of the relative convex hull of T . Let a be such a point that is closest to l . Then, there exists a path in T from v to a that remains in S_i . In another words, for each point q

not in S_i , there is a point x in T such that q and x are on opposite sides of u on l , implying $q \notin V(T, H_i)$.

Case 1:

Case 2: In this case, the supporting line l of e intersect T in two points a and b on opposite sides of e on l . Therefore, by the same argument used above, every point q not in S_i is also not in $V(T, H_i)$.

Case 2

Case 3: In this case, the supporting line l of e must intersect T at point a . Then l must also intersect H_i in some point b between e and a on l . Thus, the existence of a path from u to a that remains in S_i shows that any point q not in S_i is also not in $V(T, H_i)$.

Case 3:

Lemma 5.3.3: Let $CV(P, TV)$ be the complete visibility polygon of P from T with respect to the set of polygonal holes I inside of CT , the total number of vertices of $CV(P, TV)$ is at most n .

Proof: Polygon P with the set of holes has n edges in total, each edge of the hole or the boundary of P can intersect at most one lid of T , resulting in at most n edges inside CT . Each edge's supporting line bounds one half-plane. The intersections of all these n half-planes generate $CV(P, TV)$. So, the total number of vertices of $CV(P, TV)$ is at most n .

Corollary 5.3.1: Total number of vertices of CT is at most $(n + k)$.

Lemma 5.3.4: Computing $CV(P, TV)$ can be done in $O(n \log n)$ time

Proof: Theorem 7.10 of Preparata and Shamos [50] states that the intersection of N half-plane can be found in $O(N \log N)$ time. Lemma 5.3.3 has proven that $CV(P, TV)$ has at most n half-planes, so the running time for computing $CV(P, TV)$ is $O(n \log n)$ time.

5.3.4.2 Eliminate holes from polygon P

By making polygon T convex, we have generated a list O , which contains the set of holes that are outside of CT . At this stage, we are looking into how to eliminate holes from polygon P to simplify the configuration. This can be accomplished by using Heffernan and Mitchell's [32] $O(n + h \log h)$ time algorithm. The algorithm computes the visibility polygon from a point among a set of h pairwise-disjoint polygonal holes with a total of n vertices. The result is a hole-free visible polygon from a point x , denoted by $CV(P \setminus O, x)$. This complete visibility polygon is outside of T and contains no holes. By doing this, we reduce our problem from computing a complete visibility polygon from a polygon **with** holes to computing a complete visibility polygon from a polygon **without** holes. This step significantly simplifies our algorithm.

Lemma 5.3.5: Let Polygon PV be a subset of polygon P without containing any holes inside CT . Let $CV(PV, x)$ be the complete visibility polygon of PV from a point x . The total number of vertices of $CV(PV, x)$ is at most $(2n + 2h)$.

Proof: Each edge of the boundary of P and the set of holes can intersect at most one lid of T , resulting in a total of $2n$ vertices in PV . In addition, each sub-polygonal hole contributes two supporting lines to merge with the boundary of PV , giving a total of $2h$ vertices in the boundary of $CV(PV, x)$. Therefore, total number of vertices of $CV(PV, x)$ is at most $(2n + 2h)$.

Figure 5.3.7: An example of PV

Figure 5.3.8: An example of Lemma 5.3.5

Lemma 5.3.6: Let Polygon $P \setminus O$ be a subset of polygon P without containing any holes outside CT , and let $CV(P \setminus O, CT)$ be the complete visibility polygon of $P \setminus O$ from CT . The number of vertices of $CV(P \setminus O, CT)$ is at most $4(n + h)$.

Proof: Once we eliminate the holes from polygon P and make the polygon T convex, polygon P is reduced to $P \setminus O$, and T becomes CT . We can then apply Ghosh's algorithm to compute $CV(P \setminus O, CT)$ in $CV(P \setminus I, x)$. The algorithm visits all vertices of P twice (for the implementation details of the algorithm, refer to Chapter 3). Polygon $CV(P \setminus O, CT)$ is the result of all intersections of tangent lines with respect to polygon CT . There are at most $(2n + 2h)$ vertices in $CV(P \setminus I, x)$ as proven in Lemma 5.3.5. Therefore, there are at most $4(n + h)$ tangent lines to generate $CV(P \setminus O, CT)$, resulting in $4(n + h)$ vertices in $CV(P \setminus O, CT)$.

Lemma 5.3.7: Computing $CV(P \setminus O, CT)$ can be done in $O(n + h + k)$ time

Proof: The complete visibility polygon of polygon $CV(P \setminus I, x)$ from CT generates $CV(P \setminus O, CT)$. Computing $CV(P \setminus O, CT)$ in $CV(P \setminus I, x)$ can be done by using Ghosh's algorithm. The algorithm runs in $O(N + K)$ time, where N is the number of vertices of the outside polygon, and K is the number of vertices of the inside polygon. In our case, the outside polygon is $CV(P \setminus I, x)$, which has $O(n + h)$ vertices as proven by Lemma 5.3.5. The inside polygon is CT , which has $O(n + k)$ vertices according to Corollary 5.3.1. Therefore, it takes total of $O(n + h + k)$ time to compute $CV(P \setminus O, CT)$.

Stage 1 of the algorithm has computed the region for all single source points from which the entire polygon T is visible without being blocked by any obstacle **inside** of CT . Stage 2 of the algorithm has computed the region for all single source points from which the entire polygon T is visible without being blocked by all obstacles **outside** of CT . Now, we have to find the subset of visible regions that sees polygon T without being blocked by all obstacles inside P .

Figure 5.3. 9: An example for stage 3 of the algorithm

a) A configuration of the example

b) Stage I result: $CV(P, TV)$, where T is visible without being blocked by holes inside CT

c) Stage 2 result: $CV(P \setminus O, CT)$, where T is visible without being blocked by holes outside CT

Stage 3: Compute $CV(P, T)$

Lemma 5.3.8: Intersection of $CV(P, TV)$ and $CV(P \setminus O, CT)$ is $CV(P, T)$.

Proof: We show that a point $p \in CV(P, T)$ if and only if $p \in CV(P \setminus O, CT)$ and $p \in CV(P, TV)$.

If $p \in CV(P \setminus O, CT) \cap CV(P, TV)$, then clearly $p \in CV(P, T)$. If $p \notin CV(P \setminus O, CT)$, a line segment (p, q) where $q \in T$ can be drawn such that pq intersects at a point x of R_e , where p does not see x . This implies $p \notin CV(P, TV)$ and $p \notin CV(P, T)$. This completes the proof.

Lemma 5.3.9: Computing $CV(P, TV) \cap CV(P \setminus O, CT)$ takes $O(n + h)$ time

Proof: Every point in $CV(P, TV)$ sees polygon T , and every point in $CV(P \setminus O, CT)$ sees convex hull of T . By definition, $T \subseteq CT$. Therefore, the intersection of $CV(P, TV)$ and $CV(P \setminus O, CT)$, is a polygon itself, which is $CV(P, T)$. It is the locus of all points from which the entire polygon T is visible without being blocked by any holes inside of P . It can be computed in $O(N)$ time, where N is the total number of vertices of two polygons. Polygon $CV(P, TV)$ has at most $2(n + h)$ vertices by Lemma 5.3.3, and $CV(P \setminus O, CT)$ has at most $4(n + h)$ vertices by Lemma 5.3.6. Therefore, computing $CV(P, TV) \cap CV(P \setminus O, CT)$ takes $O(n + h)$ time.

5.3.4.3 Put it all together

Algorithm 5.3 for computing the complete visibility polygon of P with holes from polygon T :

- Stage 1: Eliminate holes from CT
- 1. Determine the convex hull of polygon T , denoted as CT
- 2. Determine all intersections among boundary of P , CT and the set of holes. Compute lists of I and O .
- 3. For each element of I ,
 - Compute the relative convex hull R_i of the C_i or H_i with respect to T .
 - Determine all S_i for each edge of the H_i .
- 4. Compute the intersections of all S_i (Compute $CV(P, TV)$)

• Stage 2: Eliminate holes from polygon P

5. Arbitrarily choose a point x inside T
6. Compute $CV(P \setminus I, x)$
7. Compute $CV(P \setminus O, CT)$ in $CV(P \setminus I, x)$

• Stage 3: Compute $CV(P, T)$

8. Compute $CV(P \setminus O, CT) \cap CV(P, TV) = CV(P, T)$
9. Return $CV(P, T)$

Theorem 5.3: Algorithm 5.3 computes a complete visibility polygon of P with a set h of holes and n vertices from a polygon T with k vertices in $O(n \log n + n \log k + k)$ time.

Proof: The correctness of the algorithm follows from Lemma 5.3.1, 5.3.2 and 5.3.8.

Finding the convex hull of T in Step 1 can be done in $O(n)$ time by using Graham's [50] linear time algorithm.

Computing lists of I and O in step 2 can be done $O(n \log n)$ time. Theorem 7.7 of Preparata and Shamos [50] proved that the K intersections of a set of N line segments could be reported in time $O((N + K) \log N)$ time. Polygon P with a set of holes has total of n line segments, and can have at most n intersections to CT . Therefore, Step 2 takes $O(n \log n)$ time.

Computing the relative convex hull R_i of C_i or H_i with respect to T in step 3 can be done using point location technique. Taking one vertex of each sub-polygon inside CT to test for point location in CT , it can be done in $O(\log(\text{size of } CT))$, which is $O(\log(n + k))$ by Corollary 5.3.1, there are at most $n/2$ polygons inside of CT , therefore, we can complete Step 3 in $O(n \log(n + k)) = O(n \log n + n \log k)$ time.

Computing $CV(P, TV)$ in Step 4 takes $O(n \log n)$ time by Lemma 5.3.4.

Arbitrarily choosing a point in Step 5 takes $O(1)$ time.

Computing $CV(PV, x)$ in Step 6 takes $O(n + h \log h)$ time by using Heffernan and Mitchell's algorithm [32].

Computing $CV(P \setminus O, CT)$ in Step 7 takes $O(n + k + h)$ time by Lemma 5.3.7.

Computing $CV(P, T)$ in Step 8 takes $O(n + k)$ time as proven in Lemma 5.3.9.

Therefore, the total running time of the algorithm is $O(n + n \log n + n \log k + h \log h + h + k) = O(n \log n + h \log h + n \log k + k) = O(n \log n + n \log k + k)$ time.

5.4 Problem 4: If T is not visible from a single point, then, compute the minimal set of points that sees T .

Specification

Given

- A polygon P with a set h of holes having a total of n vertices
- A polygon T inside polygon P with k vertices

Compute

- Determine if the k -radiation source points is NP -hard.

We claim that finding the minimum set of points that sees T is an NP -hard problem.

Recall Chapter 4. We have discussed the complexity classes: P and NP . Problems in P can be solved in deterministic polynomial time, and problems in NP can be solved in Non-deterministic polynomial time. The hardest problems in NP , the NP -complete problems, appear to be intractable: despite the facts that many NP -complete problems have been identified, no polynomial algorithm is known for any of them.

To proof a problem B is NP -hard, a polynomial reducibility has to be established between problem A and problem B where A is known to be an NP -complete problem. Polynomial reducibility can be formulated as follows: If problem A can be transformed in a

polynomial time to an instance of the problem B , and if we can find a solution to problem B , such that it can also transfer to be a solution to the problem A in polynomial time, then A is said to be polynomially reducible, or A is reducible, to B . If A is reducible to B , then B is at least as hard as A , for if B can be solved in polynomial time, so can A (see Figure 5.4.1).

Figure 5.4.1: Polynomial reducibility between problem A and B

A problem $B \in NP$ to which all problems, $A \in NP$, can be reduced is called *NP-complete*; it is at least as hard as any problem in *NP*. If an *NP-complete* problem B is reducible to a problem C , then C is called *NP-hard*, it is at least as hard as an *NP-complete* problem.

As the classes P and NP are defined for decision problems, which means that output of the question is only "yes" or "no", we therefore will phrase problem 5 as decision question also. Our proofs of NP-hardness will start from a known NP-complete problem, the boolean minimum vertex guard set for a polygon T . It may be stated as follow:

The minimum vertex guard set covers a polygon without hole

INSTANCE: A set of lists of integer-coordinate vertices representing a polygonal region T , and a positive integer bound M .

QUESTION: Is there a cover of T with M or fewer star subsets of T , i.e., do there exist star polygons $S_1, S_2, \dots, S_m$ with $m \leq M$ such that $S_1 \cup S_2 \cup \dots \cup S_m = T$?

The minimum set of radiation source points that covers polygon T is formally stated as follow:

INSTANCE: A set of lists of integer-coordinate vertices representing a polygonal region T , and a set of size of H disjointed polygons outside of polygon T , which represents the set of holes, and a positive integer bound M , where $H \leq M$.

QUESTION: Is there a cover of T with M or fewer star subsets of T , i.e., do there exist star polygons $S_1, S_2, \dots, S_m$ with $m \leq M$ such that $S_1 \cup S_2 \cup \dots \cup S_m = T \setminus H$?

The goal is to polynomially transform a given instance of the minimum vertex guard for a polygon T problem into the minimum set of radiation source points problem.

Before we can show how the vertex guard problem can be polynomially transformed into the minimum set of radiation source points problem, a few assumptions have to be stated:

1. The set of holes is always outside of the polygon T .
2. The radiation source points have to be outside of the polygon T and the set of holes.

Also, they can not be moved between a hole and the polygon T (see Figure 5.4.2).

3. We do allow the radiation sources to be on the tumor (the polygon T) if they are not blocked by the set of holes (see Figure 5.4.2).

Figure 5.4.2: Restriction on the movement of a radiation source point

Lemma 5.4.1: The minimum set of the radiation source points solution has to be a subset of the vertices of the polygon T .

Proof: Given an instance of the minimum guard set problem, we have a polygon T with m vertices. By adding a set of holes along the edges of T , we transform the given instance into the minimum set of radiation source points problem (see Figure 5.4.3).

Figure 5.4.3: Adding a set of holes along the edges of polygon T

In order to cover the polygon T , the radiation source point can be placed anywhere outside of T and the set of holes. It could be on a vertex, or a point on an edge (but can not be moved along the edge) where it was not blocked by a hole. Take a finite segment l inside the polygon T , we notice that for any source point that is not on the vertex of T , it sees a point or a portion of l , but it may not be able to see the entire segment of l . An example is shown in Figure 5.4.4. Source point B only sees partially the line segment l .

Since l is a finite segment with infinite points, we conclude that the finite solution for the minimum set of radiation source points can not be on those non-vertex points (see Figure 5.4.4). Conversely, if the source point is on a vertex of T , it could see the entire segment (see Figure 5.4.5). We therefore complete our proof.

Figure 5.4.4 Source points that are not on vertices of T

Source A sees only a point of l
Source B sees partial of l

Figure 5.4.5: Source points on vertices of T

Source at point A sees the whole segment l ,
 Source at point B sees only partial segment of l

We now show that a vertex guard problem is polynomially transformable to a minimum radiation source points problem.

Lemma 5.4.2 Constructing a set of holes outside the polygon T requires only polynomial time.

Proof: Given an instance of the minimum guard set of a polygon T with m edges, we first can build a set of m holes along each edge of T . Each hole needs to be either horizontal or vertical to the vertex. It defines for each vertex a tiny strip where the radiation source can be. Since this step visits the edges of the polygon T only once, it therefore needs only linear time to complete the size of m holes outside of T . In some cases, it is possible that

some of the strips are intersected as we show in Figure 5.4.6. Placing radiation sources on those intersection points, which are not on vertex of T , does not yield to an optimal solution as we have proven on Lemma 5.4.1.

Figure 5.4.6: One radiation source point sees more than one place

On the other hand, we can move those non-vertex source points to the corresponding vertex because there is no hole that will block us to do that (see Figure 5.4.7). Then, all possible radiation source points are on the vertices of the polygon T . It guarantees that the minimum radiation source could only be a subset of those vertices. We completed our reduction algorithm.

Figure 5.4.7: Moving the radiation sources to be on the vertices

Theorem 5.4.1: Determining the minimum set of points that covers the polygon T is an NP-hard problem.

Proof: Lemma 5.4.1 and 5.4.2 establish that a minimum vertex guard for a polygon T is polynomially transformable to a minimum radiation source points for a tumor T with a set of holes. Since the minimum vertex guard is known to be an NP-hard problem, therefore, finding the minimum set of points that covers polygon T is also an NP-hard problem.

Given a polygon P with n vertices, Ghosh [27] found an approximation algorithm for finding a vertex guard set in $O(n^5 \ln n)$ time that has at most $O(\ln n)$ times the minimum number of vertex guards needed to guard P . If k -radiation source points are placed on vertices only, then Ghosh's approximation algorithm can be applied for solving problem 4.

Chapter 6 Summary and Open Problems

6.1 Summary

Although computers have been used for the radiation therapy treatment planning, due to the large volumes of data required to be processed, a more sophisticated and efficient computer control treatment planning system is needed. Providing a better algorithm for locating x-ray source is one way to improve the treatment planning system. As we have stated at the beginning of the thesis, the x-ray source location problem can be translated to a computational geometric problem. It involves finding the visibility region in a given geometry configuration.

The first topic we have examined was algorithms for computing visibility polygons from a given source. A given source can be a point, a line segment or a polygon. It can be inside a polygon with or without holes, or outside of a polygon. We have found out that computing the visibility from a polygon inside a polygon with holes was an open problem. However, Ghosh has presented an algorithm for computing the visibility polygon from a polygon inside a polygon without holes. We have made use of this algorithm into Algorithm 5.3. We gave a detailed description of the algorithm's operation, and more importantly, we have discussed its impact to algorithm 5.3 in Chapter 3.

In Chapter 4, we have switched from the study of the algorithms for computing visibility polygons from a given source to the study of the Art Gallery theorems and algorithms' research. It guides us to know if we can find an algorithm for computing the minimum set of x-ray source points to cover the radiation target area in polynomial time.

Chapter 5 is the main part of the thesis. In this chapter, we have presented five algorithms for the radiation source location problem. When the target volume, surrounding organs and critical structures are well separated, oncologists may select the x-ray source point based on the CT or MRI scans and input it to a treatment-planning computer. Algorithm 5.1 can be applied to verify if the visibility region from the given point contains the entire tumor area. Algorithm 5.2 provides information for oncologists to visualize how much area a given point can cover. It will be useful when the tumor area and the surrounding critical structures are not well separated geometrically and multiple source points are needed to target the tumor area. Oncologists need to know if any two visible regions from the x-ray source points overlap, which is called "cross-firing area". If so, care has to been taken when performing the dose calculation to make sure no over dose on those cross-firing areas while the target area receives the proper dose and the nearby health issues receive only minimum doses. Algorithm 5.3 provides an optimal way to find all possible source points that can cover the entire tumor. The result of the Algorithm 5.3 could be a set of possible points or an empty set. In the first case, it is up to the oncologists to decide which point(s) will best suit for the treatment. In the latter case, it indicates that no any single point can cover the entire tumor without a passthrough to the critical areas. It may

need two or more source points to cover the entire target area. However, we have proved that finding the minimal set of points to cover the entire tumor is a NP-hard problem as we have discussed in Algorithm 5.4. We therefore could only find an approximate solution.

6.2 Open Questions

A better radiation therapy system will depend on a better treatment planning system. It is important to develop a more sophisticated and more efficient treatment planning system. However, a lot of questions need to be answered before we can build a next generation treatment planning system. For example, does there exist an arc such that a mobile treatment machine can move along to cover the entire tumor? Given a constant m , where m represents numbers of cross-firing areas, does there exist a set of points that their cross-firing areas cover the entire tumor but the numbers of cross-firing at the same area do not exceed m times? Can one find solutions for the same set of problems in three-dimension? Can they achieve the optimal time complexity? Also, Chandru, Ghosh, Maheshwari, Rajan and Saluja [10] have studied parallel computation of the complete visibility polygon of a simple polygon from a set of points T inside it. They obtained an algorithm which takes $O(\log n)$ time and $O(n)$ space using $O(n)$ processors in the *CREW-PRAM* model, where n is the total number of vertices of P and T . The algorithm will inspire us to solve the radiation source location problems in parallel computing. Furthermore, we need to consider how dosage versus distance to the target affects the accuracy of the radiation

treatment, or how we can find a radiation path that the oncologist finds it acceptable and the patient feels comfortable.

References

1. M. Abellanas, J. Garcia-Lopez, and F. Hurtado "Consecutive Guards" *Proceedings of the Sixth Canadian Conference on Computational Geometry*. pp.393 – 398, 1994.
2. E. Arkin, and J.S.B. Mitchell "An optimal visibility algorithm for a simple polygon with star-shaped holes". *Technical reports 746*, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY. June, 1987.
3. T. Asano, L. Guibas. J. Hershberger, and H. Imai "Visibility of disjoint polygons". *Algorithmica 1* pp.49 – 63,1986.
4. T. Asano "Efficient algorithms for finding the visibility polygons for a polygonal region with holes". *Transactions of IECE of Japan*. E-68 pp.557-559, 1985.
5. D. Avis, and G. Toussaint. "An optimum algorithm to determine the visibility of a polygon from an edge". *IEEE Transactions Computer C-30*, pp.910-914, 1981.
6. D. Avis, and G. Toussaint "An efficient algorithm to decompose a polygon into star-shaped pieces". *Pattern Recognition 13*, pp.295-298, 1981.
7. D. Avis, and H. El Gindy "A combinatorial approach to polygon similarity" *IEEE Transactions on Information Theory*, IT-29, pp.148-150, 1983.
8. I. Bjorling-Sachs, and D. Souvaine. "A tight bound for guarding polygons with holes". *Rutgers University Technical Report LCSR-TR-165*, 1991.
9. I. Bjorling-Sachs, and D. Souvaine. "An efficient algorithm for guard placement in polygons with holes". *Discrete and Computational Geometry 13*, pp.77-109, 1995.

10. V. Chandru, S. K. Ghosh, A. Maheshwari, V. T. Rajan, and S. Saluja "NC-Algorithms for Minimum Link Path and Related Problems". *Journal of Algorithms* 19, pp. 173-203, 1995.
11. B. Chazelle, and D. Dobkin "Optimal convex decompositions" *Computational Geometry*, G.Toussaint, ed. , North Holland, pp.63-133,1985.
12. B. Chazelle, and L. Guibas "Visibility and intersection problems in plane geometry". *Proceedings of the first Annual ACM Symposium on Computational Geometry*. pp.135 – 146, 1985.
13. B. Chazelle, and L. Guibas "Visibility and intersection problems in plane geometry". *ACM Transactions of Graphics*, 4: pp.551-581, 1989.
14. B. Chazelle. "Triangulating a simple polygon in linear time". *Discrete and Computational Geometry* 6, pp.485-524, 1991.
15. D. Z. Chen, V. Estivill-Castro, and J. Urrutia "Optimal guarding of polygons and monotone chains" *Proceedings of the Seventh Canadian Conference on Computational Geometry*. pp.131 – 138, 1995.
16. V. Chvatal. " A combinatorial theorem in plane geometry". *Journal of Combinatorial Theory Series B* 18. pp.39 -41, 1975.
17. K. Clarkson, R. E. Tarjan, and C. Van Wyk "A fast Las Vegas algorithm for triangulating a simple polygon" *Discrete Computational Geometry* 4, pp. 423 - 432, 1989.
18. T.H. Cormen, C.E. Leiserson, and R. L. Rivest "Introduction to Algorithms" *The MIT Press*, Cambridge, Massachusetts London, England, 1992.

19. J. Culberson, and R. Reckhow "Covering polygons is hard" *Proceedings of the 29th Symposium on Foundations of Computer Science*, pp.601-611, 1988.
20. H. Edelsbrunner, J. O'Rourke, and E. Welzl "Stationing guards in rectilinear art galleries". *Computer Vision, Graphics, and Image Processing* 27, pp. 167-176,1984.
21. H.E. El Gindy, and D. Avis "A linear algorithm for computing the visibility polygon from a point". *Journal of Algorithms* 2, pp.186 – 197, 1981.
22. H.E. El Gindy "An efficient algorithm for computing the weak visibility polygon from an edge in simple polygons". *Technical Report*, School of Computer Science, McGill University, 1984.
23. U. Finke, and K. H. Hinrichs "Overlaying simply connected planar subdivisions in linear time". *Proceedings of the Eleventh Annual ACM Symposium on Computational Geometry*, pp.119 – 126, 1995.
24. S. Fisk "A short proof of Chvatal's watchman theorem". *Journal of Combinatorial Theory Ser. B* 24, pp.374, 1978.
25. M. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan "Triangulating a simple polygon", *Information Processing Letters* 7, pp.175-179, 1978.
26. L. Gewali "Placing guards inside orthogonal polygons". *Information Sciences* 88, pp. 1 - 14,1996.
27. S. K. Ghosh "Approximation algorithms for Art Gallery Problems", *Proceedings of the Canadian Information Processing Society Congress*, 1987.
28. S. K. Ghosh "Computing the Visibility Polygon from a Convex Set and Related Problems". *Journal of Algorithms* 12, pp.75– 95, 1991.

29. L. J. Hershberger, D. Leven, M. Sharir, and R. Tarjan "Linear time algorithms for visibility and shortest path problems inside triangulated simple polygons". *Algorithmica* 2, pp.209- 233, 1987.
30. L. Guibas, and R. Seidel: "Computing convolutions by reciprocal search". *Proceedings of the Second ACM Symposium on Computational Geometry*, pp.90-99, 1986.
31. H. D. Hecker, and D. Herwing "Some NP-hard polygon cover problems". *Journal Information Process Cybern.* 25: pp.101 - 108, 1989.
32. P.J. Heffernan, and J.S.B. Mitchell "An optimal algorithm for computing visibility in the plane". *SIAM Journal on Computing*, 24(1), pp.184-201, 1995.
33. J. Hershberger "Finding the visibility graph of a polygon in time proportional to its size". *Proceedings of the Third ACM Symposium on Computational Geometry*, pp.11-20, 1987.
34. F. Hoffmann, M. Kaufman, and K. Kriegel "The art gallery problem for polygons with holes". *Proceedings of the 32nd Symposium on Foundations of Computer Science*, pp.39-48, 1991.
35. F. Hoffman, and K. Kriegel "A graph coloring result and its consequences for the polygon guarding problems". *Technical reports B 93-08*, Freie University Berlin. 1993.
36. J. Huang, A. Maheshwari, D. Nussbaum, and J. R. Sack " A note on Approximations of Rectilinear Polygons (Extended Abstract)", *Carleton University* 1995 (see also J. Huang "Rectilinear Polygon Approximation" Honors Report Carleton university 1995).

37. B. Joe "On the correctness of a linear-time visibility polygon algorithm".
International Journal of Computer Mathematics, 32, pp.155-172, 1990.
38. B. Joe, and R. B. Simpson "Corrections to Lee's visibility polygon algorithm". *BIT*,
27, pp.458-473, 1987.
39. J. Kahn, M. Klawe, and D. Kleitman " Traditional galleries require fewer watchmen".
SIAM Journal Algebraic and Discrete Methods 4, pp. 194-206, 1983.
40. F. M. Khan "The Physics of Radiation Therapy". Published by *Williams and Wilkins*,
1993.
41. D. G. Kirkpatrick, M. M. Klawe, and R. E. Tarjan " Polygon triangulation in $O(n \log \log n)$ time with simple data-structures". *Proceedings of the sixth Symposium on Computational Geometry*, pp.34-43, 1990.
42. D. T. Lee, and F. P. Preparata "An optimal algorithm for finding the kernel of a polygon ". *Journal of the ACM* 26: pp.276-282, 1979.
43. D. T. Lee, and A. K. Lin. "Computational complexity of art gallery problems". *IEEE Transactions Information Theorem IT-32*, pp.415-421, 1979.
44. D. T. Lee "Visibility of a simple polygon". *Computer Vision, Graphics, and Image Processing*, 22, pp.207-221, 1983.
45. D. T. Lee, and A. K. Lin "Computing the visibility polygon from an edge". *Computer Vision Graphics Image Proceedings*. 34 pp.1- 19, 1986.
46. A. Lingas "The power of non-rectilinear holes". *Proceedings of the 9th Colloquium on Automata, Languages, and Programming*, Springer-Verlag, pp. 369-383, 1982.
47. S. Ntafos, and M. Tsoukalas "Optimal placement of guards". *Proceedings of the Third Canadian Conference on Computational Geometry*, pp.122 - 125, 1991.

48. J. O'Rourke "Galleries need fewer mobile guards: a variation to Chvatal's Theorem".
Geometriae Dedicata 14, pp.273-283, 1983.
49. J. O'Rourke "Art Gallery theorems and algorithms", *Oxford University Press*,
Cambridge, UK. 1987.
50. F. P. Preparata, and M. Shamos "Computational Geometry: an Introduction".
Springer, Berlin Heidelberg New York Tokyo, 1985.
51. J. R. Sack, and G. Toussaint " A linear time algorithm for decomposing rectilinear
star-shaped polygons into convex quadrilaterals". *Proceedings of the 19th Allerton
Conference on Communication, Control and Computing, Monticello*, pp. 21-30, 1981.
52. J. R. Sack, and J. Urrutia, "Handbook in Computational Geometry". *North Holland*,
2000.
53. A. Schweikard, R. Tombropoulos, L. Kavraki, J. R. Adler, J. C. Latombe "Treatment
planning for a radiosurgical system with general kinematics". *Proceedings of the
IEEE International Conference on Robotics and Automation*, pp. 1720-1727, 1994.
54. R. Seidel "A simple and fast incremental randomized algorithm for computing
trapezoidal decompositions and for triangulation polygons". *Computational Geometry
Theory Application* 1, pp.51-64, 1991.
55. T. Shermer "Visibility properties of polygons". *Ph.D Thesis*, McGill University,
Montreal, Canada, 1989.
56. T. Shermer "Recent Results in Art Galleries". *Proceedings of the IEEE*, pp.1384 -
1399, 1992.

57. S. Suri, and J. O'Rourke "Worst-case optimal algorithms for constructing visibility polygons with holes". *Proceedings of the Second Annual ACM symposium on Computational Geometry*, pp.14 – 23, 1986.
58. R. Tarjan, and C. Van Wyk. "An $O(n \log \log n)$ time algorithm for triangulating simple polygons". *SIAM journal of Computing* 17, pp.143-178, 1988.