TERRAIN VISIBILITY OPTIMIZATION PROBLEMS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL

ENGINEERING AND

THE INSTITUTE OF ENGINEERING

AND

SCIENCES OF BILKENT UNIVERSITY IN PARTIAL

FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

By

İBRAHİM DÜGER

September, 2001

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Barbaros Tansel (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Murat Fadıloğlu

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Doğan Serel

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray

Director of Institute of Engineering and Sciences

ABSTRACT

TERRAIN VISIBILITY OPTIMIZATION PROBLEMS

Düger, İbrahim

M.S. in Industrial Engineering

Advisor: Assoc. Prof. Barbaros Tansel

September, 2001

The Art Gallery Problem is the problem of determining the number of observers necessary to cover an art gallery such that every point is seen by at least one observer. This problem is well known and has a linear time solution for the 2 dimensional case, but little is known about 3-D case. In this thesis, the dominance relationship between vertex guards and point guards is searched and found that a convex polyhedron can be constructed such that it can be covered by some number of point guards which is one third of the number of the vertex guards needed. A new algorithm which tests the visibility of two vertices is constructed for the discrete case. How to compute the visible region of a vertex is shown for the continuous case. Finally, several potential applications of geometric terrain visibility in geographic information systems and coverage problems related with visibility are presented.

Keywords-Art Gallery Problem, Terrain visibility, Facility Location.

ÖZET

ARAZİ GÖZETLENMESİNİN OPTİMİZASYONU

Düger, İbrahim

Endüstri Mühendisliği Bölümü Yüksek Lisans

Tez Danışmanı: Doç. Barbaros Tansel

Eylül, 2001

Sanat Galerisi Problemi bir sanat galerisinin her noktasının en az bir kamera tarafından gözetlenebilmesi için gerekli kamera sayısının ve yerlerinin bulunması problemidir. Bu problem 2 boyut için linear bir çözüme sahip olmasına rağmen 3 boyutlu durumlar hakkında fazla bilgi bulunmamaktadır. Bu tezde, düğüm ve nokta kameraları arasındaki baskınlık ilişkisi araştırılıp gerekli düğüm kameralarının üçte biri sayısında nokta kamerası ile gözetlenebilecek bir konveks polihedronun olduğu gösterilmiştir. Kesikli durumlar için yeni bir görünürlük algoritması geliştirilip süreklilik durumları için de bir düğümün gördüğü alanın nasıl hesaplanacağı gösterilmiştir. Arazi görünürlüğü uygulamaları ve gözetleme ile ilgili kaplama problemleri de sunulmuştur.

Anahtar Sözcükler: Sanat Galerisi Problemi, Arazi Görünürlülüğü, Makina ve Tesis Yerleştirme.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

1 Introduction

The Art Gallery Problem (AGP) is the problem of determining the minimum number of cameras (observers, guards) necessary to cover an art gallery such that every point is seen by at least one camera. The AGP is posed in 1973 by Victor Klee and Chvatal [Chv75] showed that $\lfloor n/3 \rfloor$ cameras are sufficient and sometimes necessary to cover the interior of an n-sided art gallery. Subsequently Fisk [Fis78] gave a concise and elegant proof using the fact that the vertices of a triangulated polygon may be three-colored. By using three-coloring Avis and Toussaint [AT81] designed an $O(n\log n)$ algorithm for placing the cameras. Although many similar problems, including moving observers, polygones with holes and internal and external visibility have been studied in computational geometry (CG), little is known about guarding an object in three dimensions. Prosenjit Bose [Bos97] proved that $\lfloor n/2 \rfloor$ vertex guards[1] are always sufficient and sometimes necessary to guard the surface of an n-vertex polyhedral terrain and by using five coloring he presented a linear time algorithm for placing $\lfloor 3n/5 \rfloor$ vertex guards to cover a polyhedral terrain which is clearly not the optimum.

[1] A *vertex guard* is a guard that is only allowed to be placed at the vertices of terrain T. Similarly, a *point guard* is a guard that is allowed to be placed at any point on the surface of T.

Digital elevation models (DEMs) provide an abstract representation (model) of the surface of the earth by ignoring all aspects other than topography. For instance, the elevation may be specified on a set of grid points (with stipulated method of interpolation). Terrain analysis on digital elevation models (DEMs) are among the most important functions in a geographic information system (GIS) as they have many diverse applications. Of the many types of information which may be derived from digital topographic surfaces, visibility, i.e. the location and size of the area which can be seen from any given viewpoint, is especially useful for terrain analysis such as navigation, scenic lanscape assessment, terrain exploration, military surveillance and site analysis for visibility coverage on topographic surfaces. The resulting abstraction, called geometric visibility, is based only on the intersection with the terrain of the lines of sight emanating from each viewpoint. Surface attributes, vegetation, atmospheric diffraction, and light intensity are neglected.

The computation of visibility is affected by the choice of the underlying computer representation of the terrain. Only regular square grids (RSGs) and triangulated irregular networks (TINs) have been used so far.

Visibility information can be computed by a few existing methods from RSGs. In general, the visibility information generated by using RSGs is of doubtful accuracy because such information is computed using grid cells as the visibility units. It is for this reason that TINs are preferred as the model for the surface representation.

Basic analyses for visibility coverage include the determination of the number of facilities that are needed and how to locate facilities such as fire towers (for monitoring forest fires), watch towers (for military surveillance), or radio transmission stations (for television or radio broadcasting) on topographic surfaces so that the entire region can be seen or monitored. Similar problems can easily be found in many other application areas, for example, site analyses for locating a specified number of facilities (within a limited construction budget) on a topographic surface so that the area of the monitored region is maximum.

For finding the minimum set of observers on TIN, it is customary to restrict consideration to viewpoints located at vertices of the triangulation. Although this restriction makes the problem tractable, it is never questioned.

In this study, we aim to show that visibility problems involving point guards are nontrivial theoretically and present two algorithms for computing visibility and we also review some aspects of visibility problems, including Art Gallery Problems, representation of surfaces and visibility related problems.

Before giving some necessary information about surface models in Chapter 3, we first present in Chapter 2 a summary of pertinent results regarding the Art Gallery Problem. At the end of Chapter 2, we present the terrain visibility problem and give concise definitions of terrain, visibility, and also some theoretical results. In Chapter 3 digital elevation models are summarized regarding their definitions and constructions while giving their advantages and disadvantages in visibility-usage. We concentrate on triangulated irregular networks since they appear to give a more accurate representation. In Chapter 4, we ask the question "Are point guards worth-considering?" and demonstrate the effect of point guards on optimization. In visibility optimization problems, we have two given sets, the candidate set and the target set. The candidate set includes the viewpoints to be chosen to optimize the guarding and the target set includes the points to be watched. With respect to the given sets, we have 4 cases for visibility optimization problems depending on which of the sets are discrete or continuous and study two of them. In Chapter 5, an algorithm for visibility calculations is constructed when both observer and target sets are discrete. In Chapter 6, we compute the visible region of a vertex. Due to its interesting applications, we dedicate Chapter 7 to the visibility optimization problems. Finally conclusions are presented in Chapter 8.

CHAPTER 2

2 The Art Gallery Problem

2.1 Problem Definition

The Art Gallery Problem (AGP), which is posed by Victor Klee, is the problem of determining the minimum number of cameras (observers, guards) and their locations to cover an art gallery such that every point is seen by at least one camera.

A gallery is, of course, a 3-dimensional space, but a floor plan gives us enough information to place the cameras. It is customary to model a gallery as a polygonal region of n vertices in the plane. We further restrict ourselves to regions that are *simple polygones,* that is, regions enclosed by a single closed polygonal chain that does not intersect itself. Thus we don't allow holes. A camera position in the gallery corresponds to a point in the polygon. In the simplest version of the AGP, each camera is considered as a fixed point that can see in every direction, that is, has a 2π range of visibility.

A camera sees those points in the polygon to which it can be connected with an open line segment that lies in the interior of the polygon. To make this notion precise, we say that point x can see point y (or y is visible to x) iff the closed line segment xy is nowhere exterior to the polygon P, i.e. $xy \subseteq P$.

A set of cameras is said to cover a polygon if every point in the polygon is visible to some camera. Cameras themselves do not block each other's visibility.

2.2 Max over min formulation

We have now made most of Klee's problem precise, except for the phrase "How many." Succinctly put, Klee poses the problem as that of finding the minimum number of guards needed to cover any polygon of *n* vertices.

For any given polygon, there is some minimum number of cameras that are necessary for complete coverage. Thus in Figure 2.1(a), it is clear that three cameras are needed to cover this polygon of twelve vertices, although there is considerable freedom in the location of the three cameras. But that is not the worst case for all polygons of twelve vertices: the polygon in Figure 2.1(b), also with twelve vertices, requires four cameras. This is what Klee's question seeks: Express as a function of *n*, the smallest number of cameras that suffice to cover any polygon of *n* vertices.

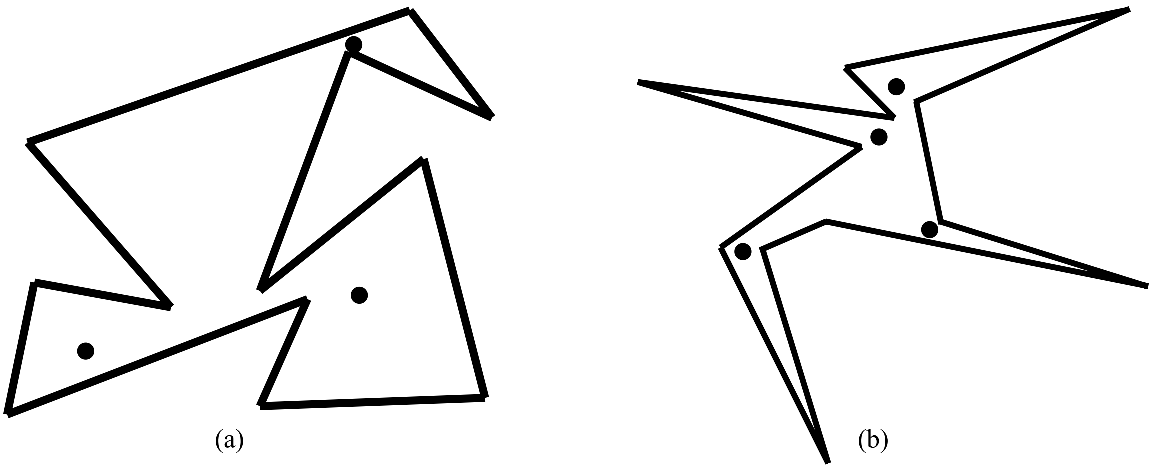


Figure 2.1 Two polygons of $n = 12$ vertices (a) requires 3 cameras, (b) requires 4 cameras

Let g(P) be the smallest number of cameras needed to cover polygon $\mathrm{P} : \mathrm{g}(\mathrm{P}) = \min_{\substack{\mathrm{S}:\,\mathrm{S}\text{ covers P} \\ \mathrm{S} \subset \mathrm{P}}} |\mathrm{S}|$, where *S* is a set of points each from *P*, and $|S|$ is the cardinality of S. Let P_n be a polygon of *n* vertices. Define G(*n*) to be the maximum of

$g(P_n)$ over all polygons of n vertices: $G(n) = \max_{P_n} g(P_n)$. Klee's problem is to determine the function $G(n)$.

2.3 Empirical Exploration

2.3.1 Sufficiency of *n*.

Certainly at least one camera is always necessary. This provides a lower bound on $G(n)$: $1 \leq G(n)$. It seems obvious that n cameras suffice for any polygon: stationing a camera at every vertex will certainly cover the polygon. This provides an upper bound: $G(n) \leq n$.

2.3.2 Necessity For Small *n*.

For small values of n, it is possible to guess the value of $G(n)$ with a little exploration. Clearly every triangle requires just one guard, so $G(3) = 1$.

Quadrilaterals may be divided into two groups: convex quadrilaterals and quadrilaterals with a reflex vertex. A vertex is called reflex if its internal angle is strictly greater than π. A quadrilateral can have at most one reflex vertex. As Figure 2.2(a) makes evident, even quadrilaterals with a reflex vertex can be covered by a single camera placed near that vertex. Thus $G(4) = 1$.

For pentagons the situation is less clear. Certainly a convex pentagon needs just one camera, and a pentagon with one reflex vertex needs only one camera for the same reason as in a quadrilateral. A pentagon can have two reflex vertices. They may be either adjacent or seperated by a convex vertex, as in Figure 2.2 (b) and (c); in each case one camera suffices. Therefore G(5) =1.

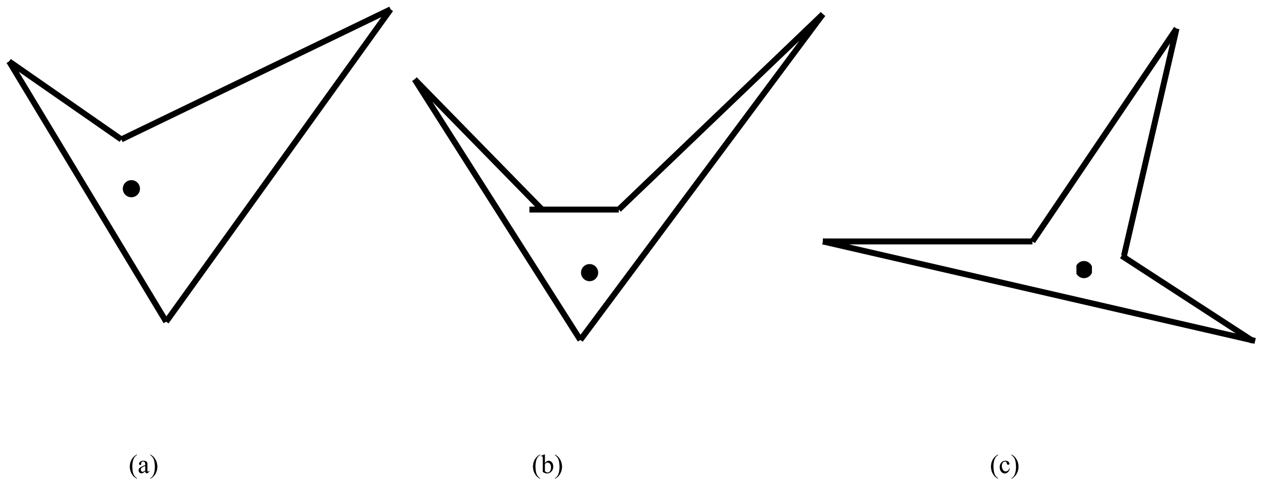


Figure 2.2: Polygons n = 4 and, n = 5 vertices.

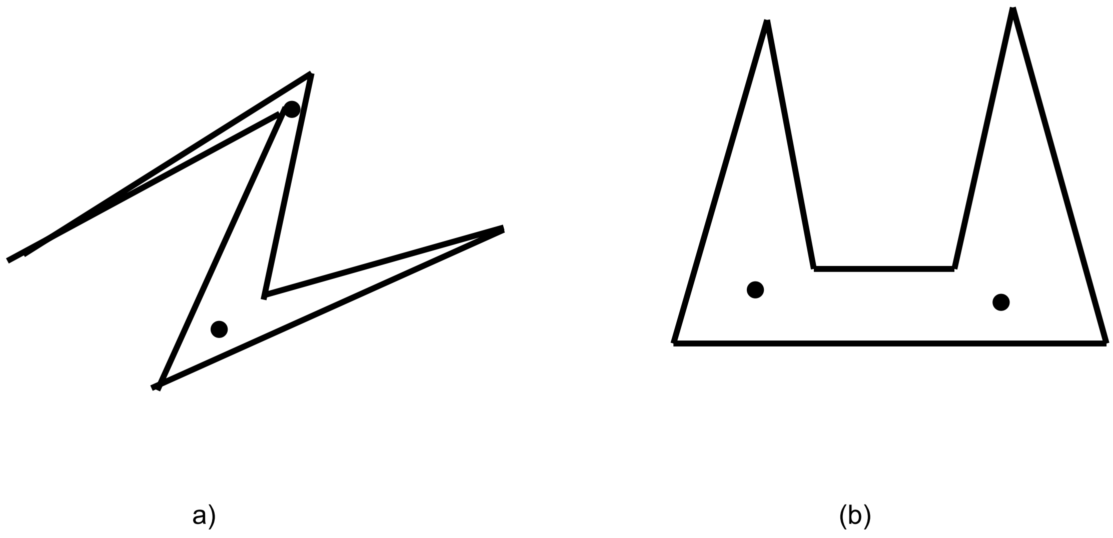


Figure 2.3: G(6) = 2.

Hexagons may require two cameras, as shown in Figure 2.3(a) and (b). A little experimentation can lead to a conviction that no more than two are ever needed, so that G(6) = 2.

2.4 Necessity of $\lfloor n/3 \rfloor$

Figure 2.4 illustrates the design for n = 12; note the relation to Figure 2.3(b). This "comb" shape consists of *k* prongs, with each prong being composed of two edges, and adjacent prongs being separated by an edge. Associating each prong with the seperating edge to its right, and the bottom edge with the rightmost prong, we see that a comb of k prongs has n = 3k edges (and therefore 3k vertices). Because each prong requires its own camera, we establish with this one example that $n / 3 \leq G(n)$ for n = 3k. Noticing that G(3) = G(4) = G(5) might lead one to conjecture that G(n) = $\lfloor n/3 \rfloor$.

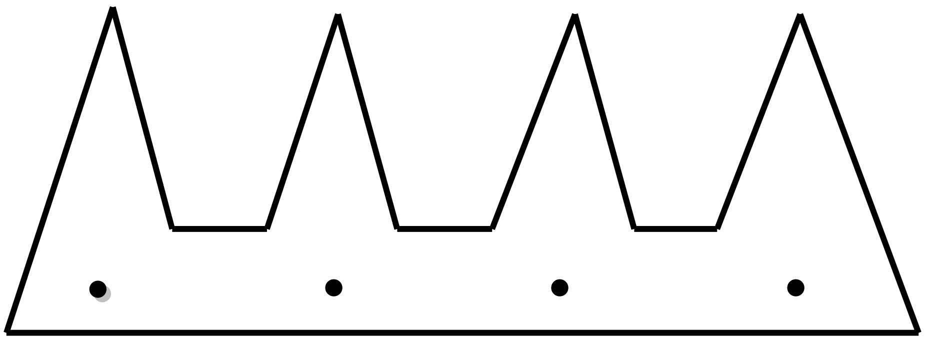


Figure 2.4: Chvátal's comb

2.5 Fisk's Proof of Sufficiency

The first proof that G(n) = $\lfloor n / 3 \rfloor$ was due to Chvátal [Chv75]. His proof was by induction: Assuming that $\lfloor n / 3 \rfloor$ cameras are needed for $n < N$, he proves the same formula for $n = N$ by carefully removing part of polygon so that its number of vertices is reduced, applying the induction hypothesis, and then reattaching the removed portion.

Three years later Fisk [Fis78] found a very simple proof, occupying just a single journal page. We will present Fisk's proof here.

2.6 Diagonals and Triangulation

Fisk's proof depends crucially on partitioning a polygon into triangles with diagonals. A diagonal of a polygon P is a line segment between two of its vertices a and b that are clearly visible to one another. This means that the intersection of the closed segment ab with, ∂P,the boundary of the polygon is exactly the set $\{a, b\}$. Another way to say this is that the open segment from a to b does not intersect ∂P except at a and b; thus a diagonal cannot make grazing contact with the boundary.

Let us call two diagonals noncrossing if their intersection is a subset of their end-points: They share no interior points. If we add as many noncrossing diagonals to a polygon as posible, the interior is partitioned into triangles. Such a partition is called a triangulation of a polygon. The diagonals may be added in arbitrary order, as long as they are legal diagonals and noncrossing. In general there are many ways to triangulate a given polygon. Figure 2.5 shows two triangulations of a polygon of *n* = 14 vertices.

2.7 Three Coloring

Assume an arbitrary polygon *P* of *n* vertices is given. The first step of Fisk's proof is to triangulate P. The second step is to " show " that the resulting graph may be 3-colored.

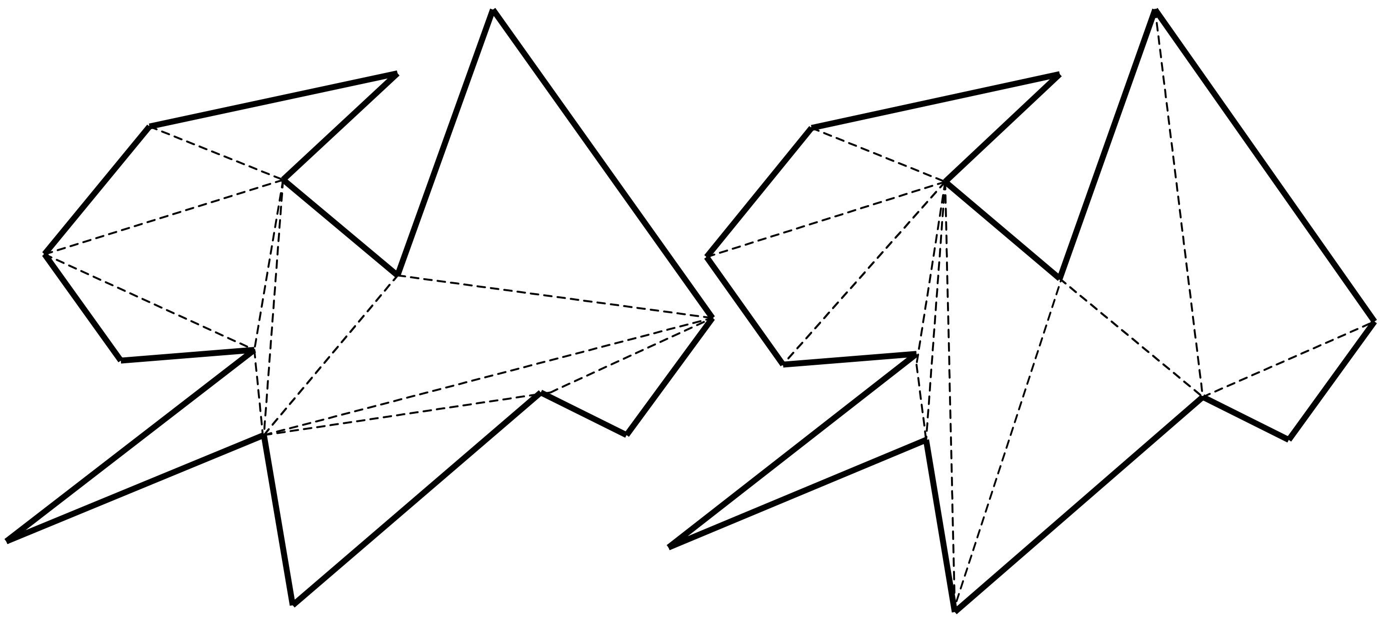


Figure 2.5: Two triangulations of a polygon of *n* = 14 vertices.

Let G be a graph associated with a triangulation such that the arcs are the edges of the polygon and the diagonals of the triangulation and the nodes are the vertices of the polygon. This is the graph used by Fisk. A *k-coloring* of a graph is an assignment of *k* colors to the nodes of a graph such that no two nodes connected by an arc are assigned the same color. Fisk claims that every triangulation graph may be 3-colored. Three coloring of Figure 2.5 are shown in Figure 2.6. Starting at, say, the vertex indicated by the arrow, and coloring its triangle arbitrarily with three colors, the remainder of the coloring is completely forced: There are no other free choices.

The third step of Fisk’s proof is the observation that placing cameras at all vertices of the same color guarantees visibility coverage of the polygon. His reasoning is as follows. Let red, yellow and blue be the colors used in the 3-coloring. Each triangle must have each of the three colors at its three corners. Thus every triangle has a red node at one corner. Suppose cameras are placed at every red node. Then every triangle has a camera in one corner. Clearly a triangle is covered by a camera at one of its corners. Thus every triangle is covered. Thus the entire polygon is covered if cameras are placed at red nodes. Similarly, the entire polygon is covered if cameras are placed at blue nodes or at yellow nodes.

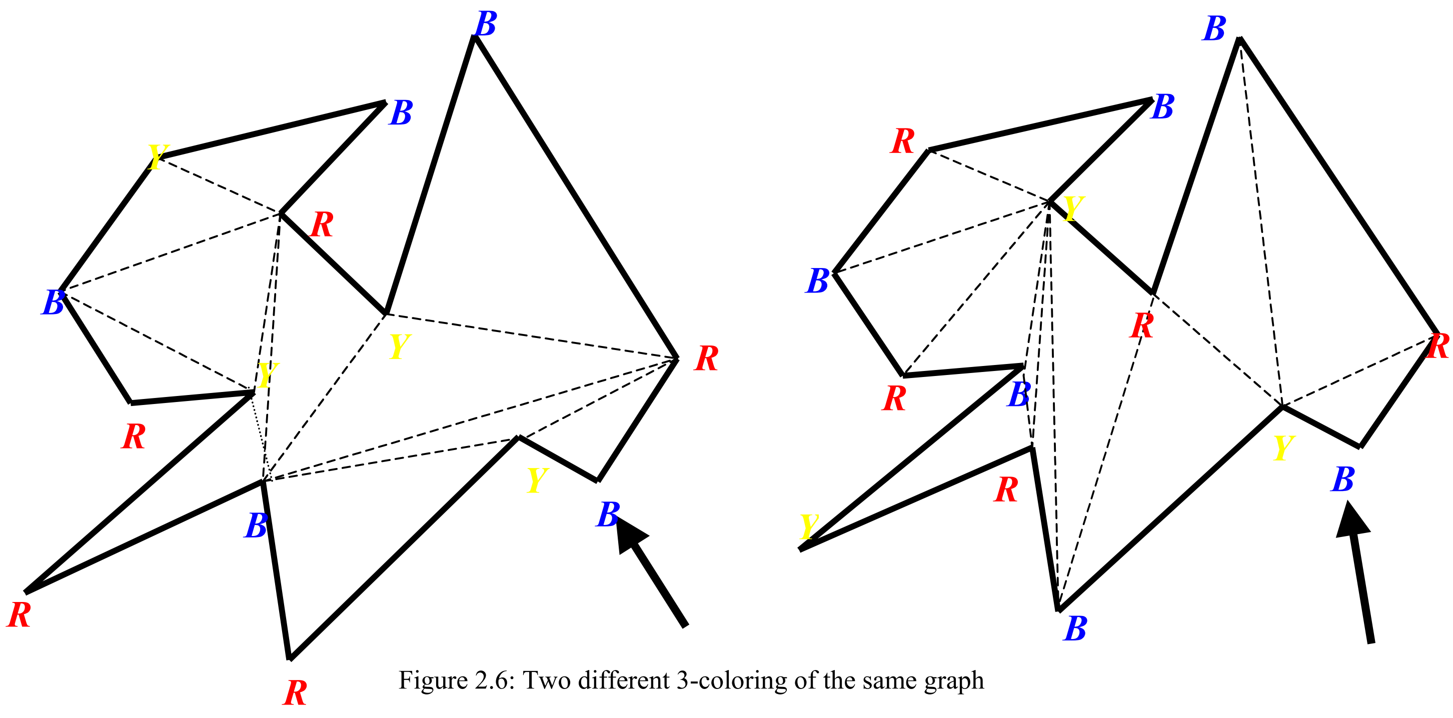


Figure 2.6: Two different 3-coloring of the same graph

The fourth and final step of Fisk's proof applies the *"pigeon-hole principle"*: If n objects are placed into k pigeon holes, then at least one hole must contain no more than n / k objects. For if each one of the k holes contained more than n / k objects, the total number of objects would exceed n. In our case, the n objects are the n nodes of the triangulation graph, and the k holes are the 3 colors. The principle says that one color must be used no more than $n / 3$ times. Since n is an integer, we can conclude that one color is used no more than $\lfloor n / 3 \rfloor$ times. We now have our sufficiency proof: Just place cameras at nodes with the least-frequently used color in the 3-coloring.

In Figure 2.6, $n = 14$, so $\lfloor n / 3 \rfloor = 4$. In (a) of the figure, yellow is used four times; in (b), the same color is used three times. Note that the three coloring argument does not always lead to the most efficient use of cameras.

By using three-coloring Avis and Toussaint [AT81] designed an O($n$log$n$) algorithm for placing the cameras. Although many similar problems, including moving observers, polygones with holes and internal and external visibility have been studied in computational geometry (CG), little is known about guarding an object in three dimensions. For more details about the 2-D problem, its applications and solutions, see [O'R87) and [She92].

2.8 Visibility on Polyhedral Terrains

The problem of guarding a polyhedral terrain was first investigated by de Floriani, et al. [dFP+86] . They showed that finding the minimum number of guards could be done using a set covering algorithm. Cole and Sharir [CS89] showed that the problem was NP-complete. Goodchild and Lee[GL89] and Lee [Lee91] present some heuristics for placing vertex guards on a terrain which will be given in Chapter 6.

Prosenjit Bose [Bos97] proved that $\lfloor n / 2 \rfloor$ vertex guards are always sufficient and sometimes necessary to guard the surface of an n-vertex polyhedral terrain and by using five coloring he presented a linear time algorithm for placing $\lfloor 3n / 5 \rfloor$ vertex guards to cover a polyhedral terrain which is clearly not the optimum. We will now give some details of this survey but first, some definitions will be given.

We define a terrain T as a triangulated polyhedral surface with n vertices $V = \{v_1, v_2,..., v_n\}$.Each vertex v_i is specified by three real numbers (x_i, y_i, z_i) which are its Cartesian coordinates and z_i is referred to as the height of vertex v_i . It is convenient to assume that z_i is non-negative so that if the X-Y plane is associated with sea-level, no points on the terrain are below sea-level. Let $P = \{ p_1, p_2 ,..., p_n \}$ denote the orthogonal projections of the points $V = \{v_1, v_2,..., v_n\}$ on the X-Y plane, i.e., each point p_i is specified by the two real numbers (x_i , y_i). It is assumed that the set P is in general position, i.e., no three points are collinear and no four are co-circular so that the projections of the edges of the polyhedral surface onto the X-Y plane determine a triangulation of P (hence the term triangulated polyhedral surface). We refer to the triangulation as the underlying triangulated planar graph associated with the terrain. Two points a, b on or above T are said to be *visible* if the line segment ab does not intersect $\overline{ab}$ any point strictly below T .

2.8.1 Vertex guards

Bose [Bos97] proves by induction that $\lfloor n / 2 \rfloor$ vertex guards are always sufficient and sometimes necessary to guard the surface of an *n*-vertex polyhedral terrain. He begins with constructing a seven-vertex graph which needs three vertex guards at least and forms the basis of the lower bound construction. Using that graph he constructs a series of planar subdivisions at each step by using the former graph. Finally he presents a linear time algorithm by using five coloring for placing $\lfloor 3n / 5 \rfloor$ vertex guards to cover a polyhedral terrain which is clearly not the optimum.

Lemma 2.1 (Lemma 3.1 in [Bos97]) The seven-vertex graph shown in Figure 2.7 needs at least three vertex guards. Furthermore, if three vertex guards are used to cover it, then at most one of the three guards can be an exterior vertex.

Proof: Suppose that two vertices suffice. One of the inner four vertices must be chosen to cover the inner triangles. If the central vertex is chosen, then the remaining unguarded (outer layer) triangles cannot be covered by one guard, as the triangles *A* and *B* do not share a vertex. Therefore, one of the three middle vertices must be chosen. Without loss of generality, suppose vertex *x* is chosen. Then, the unguarded triangles (*A* and the three triangles adjacent to *A*) are not coverable by one vertex guard.

The second step of his proof is that at most one vertex guard can be an exterior vertex. If all three were exterior vertices, then the middle three triangles would be unguarded. Suppose that at least two of the vertex guards are exterior vertices.

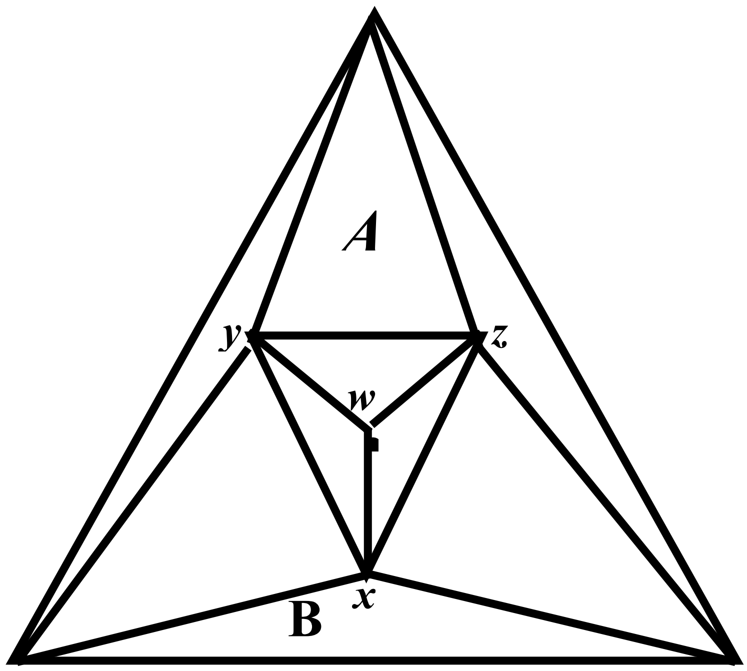


Figure 2.7. The seven vertex graph

Without loss of generality, let them be the bottom two. We now have *A* and the three central triangles (directly below *A*) unguarded. These triangles cannot be guarded with one additional guard.

Using the graph in Figure 2.7, he constructs a series of planar subdivisions S_1 ,..., S_k , where S_1 is the graph of Figure 2.7 and S_{k+1} is obtained from S_k in the following manner: let S_{k+1} be the graph of Figure 2.7 with one of the central triangles replaced by a copy of S_k (without loss of generality, suppose it is the one below face *A*). He shows the following property about S_k :

Lemma 2.2 (Lemma 3.2 in [Bos97]) S_k is triangulated, has $n_k = 4k + 3$ vertices, needs $g_k = 2k + 1$ guards, and if it is covered by exactly $2k + 1$ guards, then at most one guard is on the exterior face.

Proof: By induction on k.

Basis: $k = 1$: Follows from Lemma 2.1.

Inductive Hypothesis: For all $k \leq t$, $t \geq 1$, S_k is triangulated, has $n_k = 4k + 3$ vertices, needs $g_k = 2k + 1$ guards, and if it is covered by exactly $2k + 1$ guards, then at most one guard is on the exterior face.

Inductive Step: $k = t + 1$. S_{t+1} is triangulated by construction. It has $n_t + 4 = (4t + 3) + 4 = 4(t + 1) + 3$ vertices. He then shows that it requires $2(t + 1) + 1 = 2t + 3$ guards, and that if it uses exactly $2t + 3$ guards, only one exterior vertex is a guard. In S_{t+1} , there is a copy of S_t . By induction, this copy of S_t must use at least $2t + 1$ guards. He considers cases based on how many guards this copy of S_t uses.

Case 1: The copy of S_t uses exactly $2t + 1$ guards. Then the copy of S_t has at most one guard on one of its exterior vertices. There are 4 cases: no guard is placed on the exterior of S_t , left vertex (y) is a guard, right vertex (z) is a guard, and the lower vertex (w) is a guard.

Case 1.1: No guard is placed on the exterior of S_t . Since S_t is already covered, two guards suffice to cover the remainder of S_{t+1} . We have that $g_{t+1} = (2t + 1) + 2 = 2(t+1) + 1$. If exactly 2 guards are used, then at most one of them can be on the exterior of S_{t+1} .

Case 1.2: A guard is placed at y. This configuration requires at least 2 guards. If covered with exactly two guards ($(2t + 1) + 2 = 2t + 3$ guards total), then at most one is on the exterior face.

Case 1.3: A guard is placed at z. This case is symmetric to Case 1.2.

Case 1.4: A guard is placed at w. There is a ring of six triangles that requires two guards and at most one of these guards is on the exterior face.

Case 2: The copy of S_t uses exactly $2t + 2$ guards. Then the copy of S_t may have guards on all three of its exterior vertices (i.e. w, y, z). However, this still leaves one face (B) uncovered, so one more guard is required. If only one more guard ($2t+3$ total) is used, then only that guard may be on the exterior face.

Case 3: The copy of S_t uses more than $2t +2$ guards. Then the induction hypothesis is true.

Theorem 2.1 (Theorem 3.1 in [Bos97]) There exists a terrain on n vertices, for any $n \equiv 3 \pmod 4$ that requires $\lfloor n / 2 \rfloor$ vertex guards.

Proof: Follows directly from Lemma 2.2. For that terrain, we have:

$$g_k = 2k + 1 \text{ and } n_k = 4k + 3, \text{ therefore}$$

$$g_k = 2\,((n_k - 3)/4) + 1 = ((n_k - 3)/2) + 1 = \lfloor n_k / 2 \rfloor.$$

Theorem 2.2 (Theorem 3.2 in [Bos97]) $\lfloor n / 2 \rfloor$ vertex guards are always sufficient and sometimes necessary to guard the surface of an arbitrary terrain T with n vertices.

Proof: First 4-color the vertices of T'. This can always be done since $T'$is a planar graph. By the pigeon hole principle, among the 4 colors there must be 2 colors such that no more than $\lfloor n / 2 \rfloor$ vertices are colored by these two colors. Furthermore, these $\lfloor n / 2 \rfloor$ vertices are sufficient to guard all of the faces of T' (because every triangle must have at least one vertex colored with one of these 2 colors). Necessity follows from Theorem 2.1.

2.8.2 An Algorithm for Placing Terrain Vertex Guards

Observation 2.1 (Observation 4.1 in [Bos97) Given a five coloring of the vertices of any terrain, any set of three color classes provides a vertex guarding of the terrain since every face of the terrain is a triangle except possibly the outer face (i.e. the outer face of the underlying planar graph which need not be guarded). Based on this observation, Bose constructs a simple linear time algorithm as follows:

1. 5-color the vertices of the planar triangulation graph;
2. Among the 5 colors, choose 3 colors which are minimally used.

CHAPTER 3

3 Representation of Surfaces

A natural terrain can be described as a continuous function z = f(x,y) defined over a simply connected subset D of the x-y plane. Thus a Mathematical Terrain Model (MTM), which we simply refer to as a *terrain*, can be defined as a pair M = (D,f). The notion of a *digital terrain model* (DTM) characterizes a subclass of MTM's which can be represented in a compact way through a finite number of data. Elevation data are acquired either through sampling technologies (on-site measurements or remote sensing: tachcometers, photogrammetry: stereo pairs of air photos etc.), or through digitization of existing contour maps. Raw data come in the form of elevations at a set of points, either regularly distributed, or scattered on a two-dimensional domain; chains of points may form polygonal lines, approximating either linear features, or contour lines.

3.1 Terrain Models

There are three commonly used approaches to the digital representation of an arbitrary surface. This process of representation is referred to as the construction of a digital elevation model or DEM. Since the commonest representation of topography on a paper map is by *contours*, or lines connecting points of equal elevation, we might simply digitize the contour lines as ordered sets of points, and assume adjacent pairs of points within each line to be connected by straight lines. Contour maps are easily transposed onto paper and best understood by humans (Figure 3.1), but are not suitable for performing complex computer-aided terrain analyses. This is due to the complete lack of information about terrain morphology between two contour lines. Thus the major disadvantage of this approach as a digital representation is that it provides a very uneven density of information; uncertainity about a randomly chosen point's elevation is zero on each contour line, and increases directly with the point's distance from the nearest line. To obtain an accurate representation of an entire surface it is therefore necessary to use a large number of contours for very small intervals of elevation.

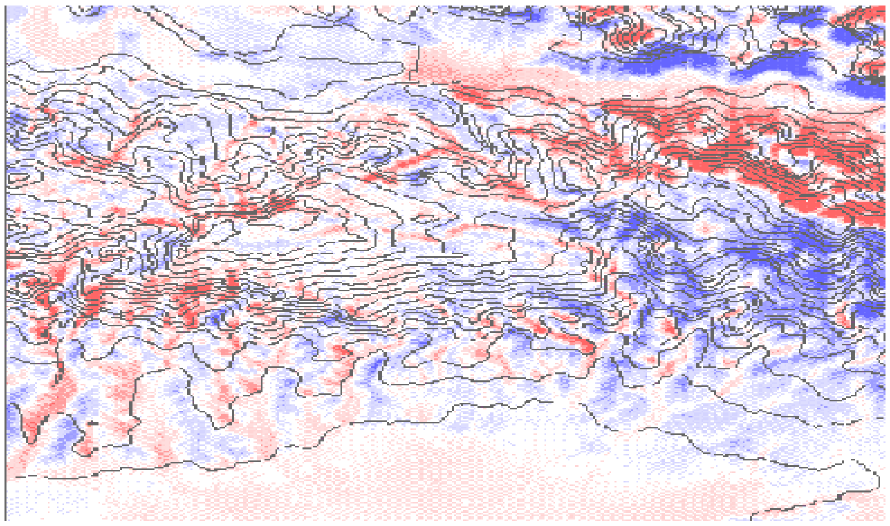


Figure 3.1. A contour map

The second alternative, the representation of the surface by a *regular square grid* of sample elevations (RSG) is a digital grid (raster or bitmap are synonyms) containing numeric values describing topography (Figure 3.2). Elevation grids are divided into rows and columns which define *grid cells* or *pixels*. Each individual pixel represents a square area on the earth's surface and contains 3 numeric values (x, y, and z) which define that pixel's column, row, and average elevation; i.e., its location in 3-dimensional space. RSG gives a uniform intensity of sampling, and is therefore frequently used in practice. Clearly it would be possible to calculate the area seen from each of the grid of sample points by interpolation.

	1	2	3	4	5	6	7	8	9	10
1	3	3	1	1	4	5	3	5	4	3
2	2	2	4	6	7	8	8	7	5	3
1	1	4	7	10	10	9	10	8	6	2
4	1	5	10	9	10	9	9	7	3	3
5	2	3	8	10	10	10	9	4	5	4
6	2	5	9	9	10	9	8	6	4	1
7	2	5	8	10	8	9	7	6	4	2
8	5	6	7	8	9	8	6	5	3	2
9	7	6	5	7	8	7	6	4	1	1
10	2	7	2	6	7	3	6	4	1	1

Figure 3.2: An RSG. The grid above represents an elevation raster. Columns (x values) and rows (y values) are vertically and horizontally arranged and numbered across the top and down the DEM. For this figure f(5,7) = 8.

However the ruggedness of any topographic surface tends to vary from one part of the surface to another. Some areas tend to be very smooth while in other areas elevation varies rapidly over short distances. For this reason, a uniform sampling density is inefficient compared to a design which responds to the variability in the surface by sampling more intensively in the more rugged areas. Moreover it is not clear how the surface should be interpolated between grid points. Any binary form (visible/invisible) of the resultant visibility information for the grid cell will be subject to “oversimplification “ as Lee claims [Lee91].

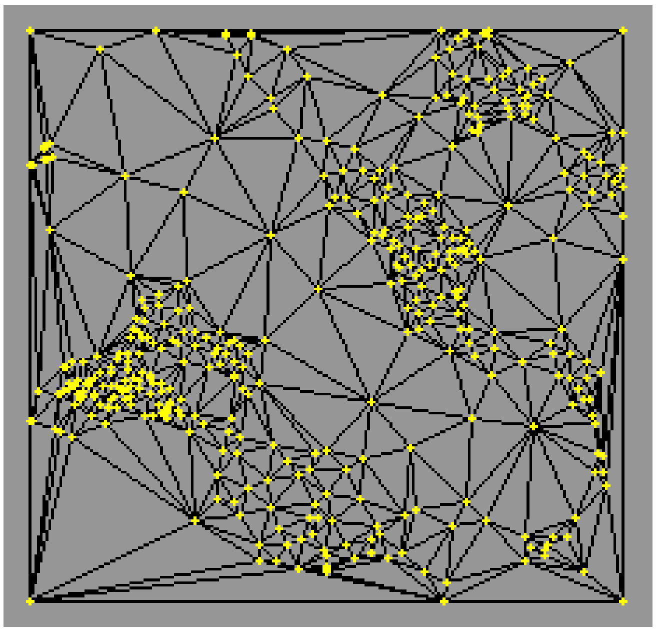


Figure 3.3. A TIN. In a TIN, the space is divided into a set of irregular triangles.

This leads logically to the third alternative, known as the TIN or Triangulated Irregular Network (see Figure 3.3). In this model, the space is divided into a set of irregular triangles with shared edges, and the surface is modeled by the triangles as if they were mosaic tiles. Each edge is shared by exactly two triangles, with the exception of those whose edges form the outer boundary of the network. Each vertex is shared by at least three triangles. In the simplest version, which is commonly used, the surface is assumed to be planar within each triangle. Advantages and disadvantages of the last two models are illustrated in [Aro89] and [Bur86].

We can triangulate the set of sample points in many different ways and there is no definitive criteria which compares different triangulations of the same sample points if we don't know the original terrain but only the heigths of it at the sample points. The construction of a TIN model begins with the selection of an irregularly located sample of points. "For maximum economy" [Lee91] the points should be more densely sampled in areas of rugged terrain. By using pits, peaks and other critical surface points on ridges and in valleys it is possible to achieve an adequate representation of a surface with far fewer sample points than with either of the previously discussed alternatives. These points then form the vertices of the TIN. One unambigious and frequently used procedure of defining the edges of the TIN is to connect all pairs of points which are Voronoi neighbors. The details of the problems about triangulations can be found in [deB97]. Due to its interesting theoretical properties and extensive preeminence, we will return to TINs in Section 3.3.

3.2 Construction and Conversion Algorithms

Raw data can come in the form of a set of points *V*, and possibly a set of lines *E*. In case the points of *V* are distributed regularly, an RSG is implicitly provided. Since sometimes MTMs are derived from contour maps, contours may also play the role of raw data. Hence, we have the following possible construction and conversion problems. Note that a clear distinction between conversion and construction techniques cannot be made since sometimes models and raw data are the same (e.g., regular distributed data points and RSGs).

1. **RSG from sparse points.** There are two possible approaches for constructing an RSG from a set V of scattered points [Pet90]:

   (a) *pointwise methods*: the elevation at each grid point p is estimated on the basis of a subset of data that are neighbors of p. There are different criteria to define the neighbors that must be considered, e.g.: the closest k points, for some fixed k; all points inside a given circle centered at p, and of some given radius; the neighbors of p in the Voronoi diagram of $V \cup \{ p \}$, etc. The basic geometric structure for all such tasks is the Voronoi diagram of the given data points.

   (b) patchwise interpolation methods: the domain is subdivided into a number of patches, which can be either disjoint or partially overlapping, and either regular or irregular shape. The terrain is approximated first within each patch through a function that depends only on data inside the patch. The elevation at grid nodes inside each patch is estimated by sampling the corresponding function.

2. **RSG from TIN.** Some systems first compute a TIN from sparse data points then they convert such a representation into an RSG [Web90]. This conversion is indeed a special case of patch interpolation methods described above.

3. **RSG from contours.** Early methods performed in this method as follows: a number of straight lines at horizontal and vertical directions are drawn through each node of the grid, and their intersections with the contour lines are computed; terrain profiles along each line are approximated by some function interpolating contours at intersection points; the elevation at grid node is computed as an average of its approximate elevations along various profiles. The underlying geometric problem is to find intersections between contour map, and the lines through each grid node. More recent approaches perform this conversion in two steps: contours are first converted into a TIN, then such a TIN is converted to produce the final RSG . See [Pet90] for details.

4. **TIN from points.** A TIN is obtained from sparse data points by computing a triangulation having vertices at data points. In case raw data also includes line segments, a constrained triangulation is computed. See [Web90] for details.

5. **TIN from RSG.** This conversion is usually aimed at data compression: the adaptivity of the TIN to surface characteristics is exploited to produce a model of terrain that can be described on the basis of a reduced subset of elevation data from an input RSG. Hence, RSG to TIN conversion involves approximation. See [Web90] for details.

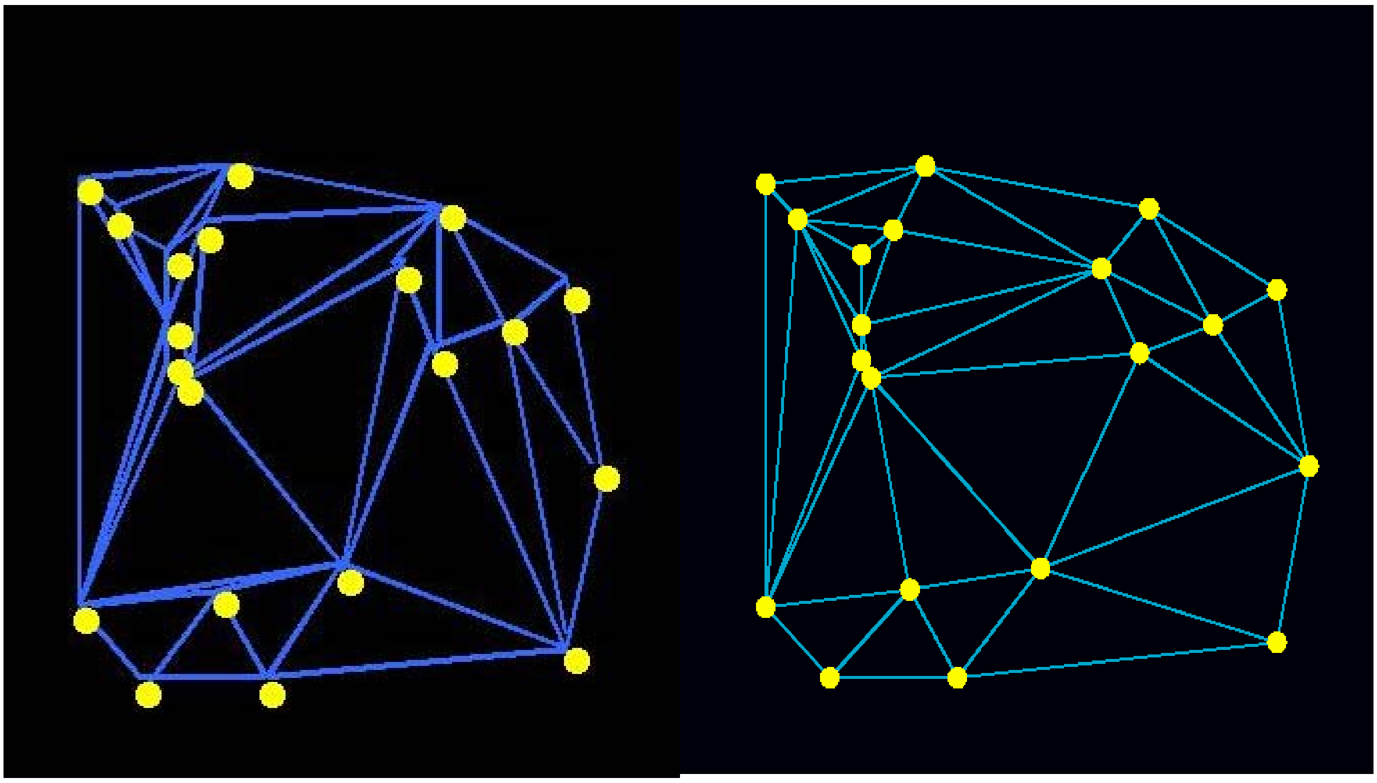
6. **TIN from contours.** A TIN conforming to a given contour map should be based on triangulation that conforms to the set of contours. This problem has been studied in the literature both in the context of GIS, and more generally in the reconstruction of three-dimensional object models. See [Web90] for details.

3.3 Triangulated Irregular Networks (TINs)

Given a finite set V of points in the plane, a *triangulation* of V is "a maximal straight-line plane graph having V as its set of vertices" ([deB97]). Thus, in a triangulation, every region, except for the external region, is a triangle. When a triangulation is taken as the basis for a digital surface model, the approximated elevation of a point P, internal to a triangle, is obtained as a function of the elevations of the vertices of that triangle.

The problem of finding an optimal triangulation of a given set of points has been considered for many different applications. In surface approximation problems, a criterion related to the size of the angles of triangles is used. A better approximation is obtained when the three vertices of the triangle lie as close as possible to P. Intuitively, a *Delaunay triangulation* of a set V of points, is, among all the possible triangulations of V, the one in which triangles are as much equiangular as possible (see Figure 3.4).

The Delaunay triangulation of a set V of points in the plane is usually defined in terms of another geometric structure, the *Voronoi diagram*. The Voronoi diagram of a set V of n points is a subdivision of the plane into n convex polygonal regions, called *Voronoi regions*, each associated with a point P_i of V. The *Voronoi region* of P_i is the set of points of the plane which lie closer to P_i than to any other point in V. Two points P_i and P_j of V are said to be *Voronoi neighbors* when the corresponding Voronoi regions are adjacent. The geometric dual graph of the Voronoi diagram is a plane graph $T \equiv (V,E)$, called the *Delaunay graph* of V, whose edges join pair of



(a) (b)

Figure 3.4: (a) An arbitrary triangulation of a point set, and (b) a Delaunay triangulation of the same set.

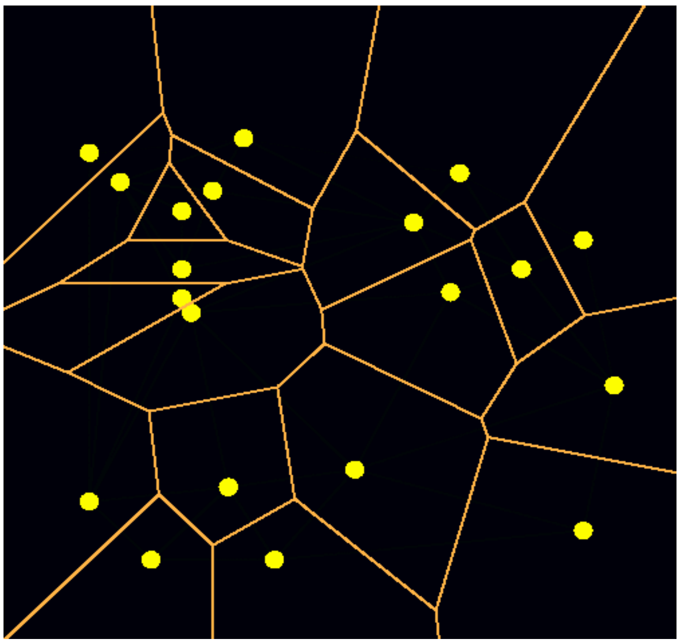


Figure 3.5: The Voronoi diagram of the same point set of Figure 3.4.

points P_i, P_j ($i \neq j$) of V, such that P_i and P_j are Voronoi neighbors (Figure 3.5). The Delaunay graph explicitly represents the Voronoi neighborhood relation induced by the Voronoi diagram over set V.

An alternative characterization of the Delaunay triangulation is given by the so-called *empty circle property.* Let τ be a triangulation of a set V of points. A triangle t of τ is said to satisfy the *empty circle property* if and only if the circle circumscribing t does not contain any point of V in its interior. A triangulation τ of V is a Delaunay triangulation if and only if every triangle of τ satisfies the empty circle property.

A Delaunay triangulation satisfies also the the *max-min angle property*, which is used operatively by several construction algorithms. Let τ be a triangulation of V, let e be an edge of τ, and Q be the quadrilateral formed by the two triangles of τ adjacent to e. Edge e is said to satify the max-min angle property if and only if either Q is not strictly convex, or replacing e with the opposite diagonal of Q does not increase the minimum of the six internal angles of the resulting triangulation of Q. An edge e, which satisfies the max-min angle property, is also called a *locally optimal edge*. A triangulation τ of V is a Delaunay triangulation if and only if every edge of τ is locally optimal.

3.3.1 Algorithms for Computing a Delaunay Triangulation

An arbitrary triangulation may not represent, in general, an acceptable solution for numerical interpolation because of the elongated shape of its triangles. Intuitively, a “good triangulation” is one in which triangles are ``as much equiangular as possible", so as to avoid thin and elongated triangular facets. Delaunay triangulation is optimal with respect to such a requirement, and thus has been extensively used as a basis for surface models.

Following de Floriani [dP92] existing algorithms for building a Delaunay triangulation or, equivalently, its dual graph (the Voronoi diagram) can be classified into the following five categories (they will be detailed after giving categories) :

- *two-step algorithms*, which first compute an arbitrary triangulation, and then optimize it to a Delaunay triangulation by iteratively applying either the empty circle or the max-min angle criteria.
- *incremental algorithms*, which construct a Delaunay triangulation by stepwise insertion of the data points, while maintaining a Delaunay triangulation at each step.
- *divide-and-conquer algorithms*, which compute a Delaunay triangulation by splitting the point set into two halves, and merging the computed partial solutions.
- *sweep-line methods*, which compute the Voronoi diagram of a set of points by first transforming it in such a way that the Voronoi region of a point P_i is considered only when P_i is intersected by the sweep-line.
- *Three-dimensional algorithms*, which compute the convex hull in 3D, and then project the lower portion on the x – y plane.

The first Delaunay triangulation algorithms were based on a *two-step strategy.* An arbitrary triangulation of the given set V of points can be obtained through the following three steps:

- sort the points of V by increasing x-coordinate;
- form a triangle with the first three non-collinear points in the sorted sequence;
- iteratively add the next point P_i by connecting P_i to all the vertices of the existing triangulation which are visible from P_i (i.e., they can be connected to without intersecting existing edges).

The optimization step iteratively applies the max-min angle (or the empty circle) criterion to any internal edge of the current triangulation, such that its two adjacent triangles form a strictly convex quadrilateral, until no more edge swapping occurs.

Incremental algorithms can be further classified into *static* and *on-line* algoritms. *Static algorithms* usually start by sorting all the points according to their euclidean distance from a fixed origin and then build the triangulation in such a way that each created triangle belongs to the final tesselation [deB97]. *On-line algorithms* are based on the incremental insertion of the internal points in an initial Delaunay triangulation of the domain. The initial triangulation of the domain can be obtained, for instance, by creating a triangle enclosing all the data points, which will be removed together with all the edges incident in its vertices at the end of the process. The update of the current Delaunay triangulation at the insertion of a new internal point P_i can be performed in an iterative approach. This approach [deB97] builds first an arbitrary triangulation by connecting P_i to the three vertices of the triangle t

of the existing triangulation, which contains P_i. The triangulation is then optimized by iteratively applying the max-min angle criterion until no more edge swapping occurs.

Divide-and-conquer algorithms perform the following four steps[O'R98]:

- the points of V are preliminarily sorted from left to right (if two points have the same x-coordinate, then the y-coordinate is considered);
- set V is split into two subsets V_L and V_R , where V_L contains the leftmost half of the points of V, and V_R, the rightmost half;
- the Delaunay triangulations of V_L and V_R are recursively constructed and then merged together to form the Delaunay triangulation of V.

The merging step of the triangulations of V_L and V_R starts with the computation of the convex hull of $V = V_L \cup V_R$, which is the domain of the Delaunay triangulation of V. This reduces to determining the lower and upper common tangent of the convex hulls of V_L and V_R (domains of the corresponding triangulations). Then, we move from the lower tangent to the upper one, by deleting the edges that are not in the final Delaunay triangulation of V, and adding the new edges.

The *sweep-line algorithm* proposed by Fortune [For87] for Voronoi diagram sweeps a horizontal line across the plane, noting the regions intersected by the line as the line moves. The algorithm computes a geometric transformation of the Voronoi diagram which has the property that the lowest point of the transformed region of a point appears at the point itself, and, thus, the Voronoi region of a point is considered only when the point itself is intersected by the sweep line.

3.4 Surface Simplification

When a large number of sampled points is available, a triangulation joining all the data can be highly inefficient in storage and for search and retrieval operations.

Approximated models based on triangular grids have been used in the past. Such models are built on the basis of a restricted subset of the data, chosen in such a way to provide a representation of the surface within a certain error tolerance. Approximated surface models are a good data compression mechanism, but they give an approximation at a predefined level of accuracy (see Figure 3.6).

The ideal aim of surface simplification is to achieve an optimal ratio between accuracy and the size representation. There are two different optimization criteria:

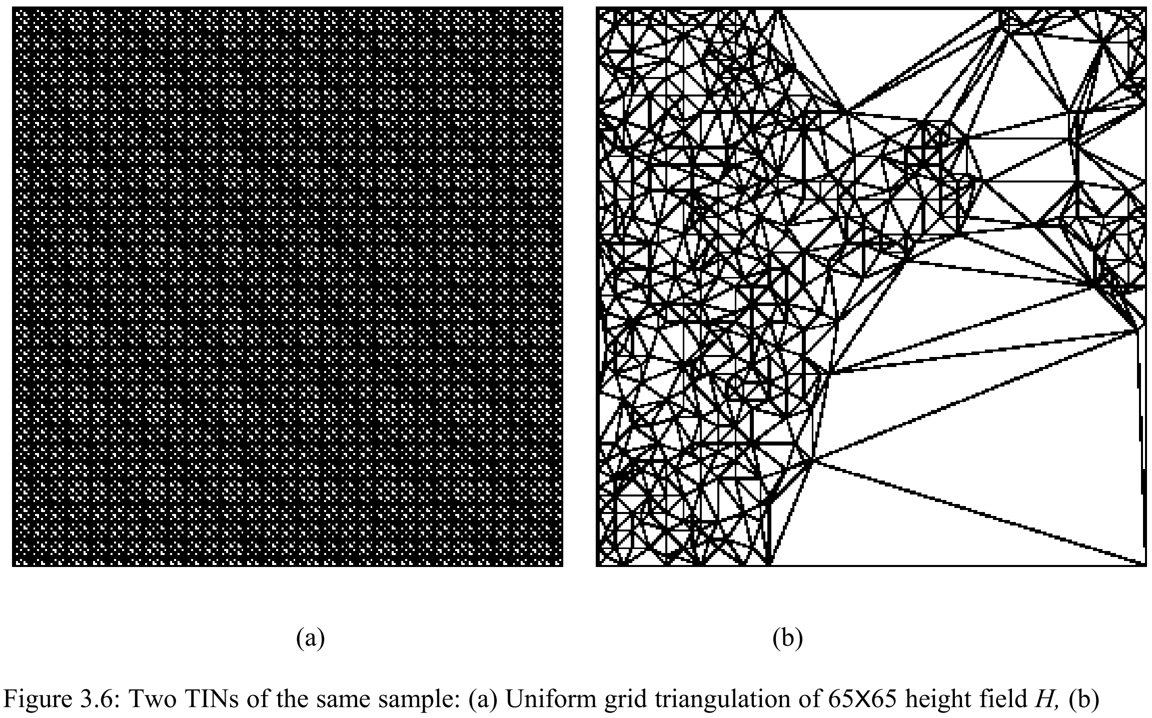

(a) (b)

Figure 3.6: Two TINs of the same sample: (a) Uniform grid triangulation of 65X65 height field *H,* (b) A triangulation τ using 512 vertices that approximates *H.*

- minimizing the number of vertices of the model for a given accuracy;
- maximizing the accuracy for a given number of vertices.

For the first problem a negative result has been proven by Agarwal and Suri [Aga94]: they consider a polyhedral terrain (for simplicity, a TIN), and show that the problem of finding an approximation for it at a given accuracy with another TIN having a minimum number of vertices at arbitrary positions is NP-hard. It was conjectured that such problems remain NP-complete even if vertices of the approximate terrain are constrained to lie at original data points.

3.4.1 Surface simplification algorithms

Only for the first problem, there exist algorithms that can achieve a suboptimal solution in polynomial time, while guaranteeing some bounds on its size. Those algorithms can be categorized into five groups (for an entire coverage of methods in detail see[HG95]):

1. *uniform grid methods*, which use a regular grid of samples in x and y;
2. *one pass feature methods*, which select a set of important \ feature" points (such as peaks,pits, ridges, and valleys) in one pass and use them as the vertex set for triangulation;
3. *multi-pass refinement methods* which start with a minimal approximation and use multiple passes of point selection and retriangulation to build up the final triangulation;

4. *multi-pass decimation methods*, which begin with a triangulation of all of the input points and iteratively delete vertices from the triangulation, gradually simplifying the approximation; and
5. *other methods*, including adjustment techniques, optimization-based methods, and optimal methods.

The latter four simplification methods typically employ two triangulation methods: Delaunay triangulation and data-dependent triangulation. *Delaunay triangulation* is a purely two-dimensional method; it uses only the *xy* projections of the input points. *Data-dependent triangulation*, in contrast, uses the heights of points in addition to their *x* and *y* coordinates (see [DLR90] for details). It can achieve lower error approximations than Delaunay triangulation, but it generates more slivers.

Within the basic framework outlined above, the key to good simplification lies in the choice of a good point importance measure. But what criteria should be used to judge such a measure? Ultimately, the final judgement must depend upon the quality of the results it produces. With this in mind, we suggest that a good measure should be simple and fast, it should produce good results on arbitrary height fields, and it should use only local information since the importance measure will be evaluated many times. Consequently, any cost inherent in the importance measure will be magnified many times due to its repetition.

3.4.2 Importance measures

We explored four categories of importance measures: local error, curvature, global error, and products of selected other measures. We briefly discuss each of these below.

- ***Local Error Measure.*** The importance of a point (x, y) is measured as the difference between the actual function and the interpolated approximation at that point (i.e. $|H(x, y) - (T\,S)(x, y)|$, where $H(x,y)$ is the height of the actual surface at the point (x,y) which is provided and $(TS)(x,y)$ is the height of the point (x,y) on the reconstructed surface). This difference is a measure of local error. Intuitively, we would expect that eliminating such local errors would yield high quality approximations, and it generally does. This measure also meets the other criteria suggested earlier: it is simple, fast, and uses only local information.
- ***Curvature Measure.*** The piecewise-linear reconstruction affected by T approximates nearly planar functions well, but does more poorly on curved surfaces. However, in everyday life, peaks, pits, ridges, and valleys, which have high curvature, are visually significant. These observations suggest that we try curvature as a measure of importance.

In one dimension, $|H''|$ is a good curvature measure. Compute values for $|H''|$ at all points and select the m points with the highest values. However, the method is over-sensitive to high frequency variations.

Because the curvature measure was inferior in one dimension, it wasn't tested in two dimensions. Laplacian, $\frac{\partial^2 H}{\partial x^2} + \frac{\partial^2 H}{\partial y^2}$, would be a good measure of curvature for functions of two variables. The Laplacian is a poor measure, however, because it sums the curvatures in the *x* and *y* directions, and these could cancel, as at a saddle. Consider $H(x; y) = ax^2 - ay^2$, for any a, for example. A better measure is the sum of the squares of the principal curvatures, which can be computed as the square of the Frobenius norm of the Hessian matrix: $\left(\left(\frac{\partial^2 H}{\partial x^2}\right)^2 + 2\left(\frac{\partial H}{\partial x \partial y}\right) + \left(\frac{\partial^2 H}{\partial y^2}\right)^2\right)$.

- ***Global Error Measure***. At every point, the global resultant error of a new approximation is formed by adding that point to the current approximation, measured as $\sum_{(x,y)} \left| H(x; y) - (T\ S)(x; y) \right|$. Then the point that produces the smallest global error is selected.
- ***Product Measures.*** Combine one or more of the importance measures given above with some bias measures. Two examples of bias measures are: absolute height, and the ratio of the number of unselected points in a region to the number of points remaining to be selected.

CHAPTER 4

4 Point Guards

To make the visibility optimization problem tractable, the search for the optimum set of observers is limited to a finite set of locations, which should include all of the peaks of the terrain. Thus the process of selection of TIN vertices has been a custom both as a means representing the surface and as a method of selecting a discrete set of locations to be searched. The validity of this method of representation has not been questioned. In this chapter we consider the problem of whether or not there exists a point guard set with smaller number than that of vertex guards to cover a polyhedral terrain. First, let's remember some definitions.

We define a terrain T as a triangulated polyhedral surface with n vertices $V = \{v_1, v_2, \ldots, v_n\}$. Each vertex v_i is specified by three real numbers (x_i, y_i, z_i) which are the Cartesian coordinates with z_i being referred to as the height of vertex v_i . A *vertex guard* is a guard that is only allowed to be placed at the vertices of T. Similarly, a *point guard* is a guard that is allowed to be placed at any point on the surface of T. G_v is the number of the vertex guards needed to cover a polyhedral terrain and G_p is the number of point guards needed.

4.1. Existency of a better point guard

In this subsection we give an example which demonstrates that an optimal vertex guard may be strictly worse than an optimal point guard. This proves that restricting the search for a guard to the vertices of a TIN may be suboptimal as compared to a search over the entire surface.

Suppose we are given 3 triangles *A*, *B*, and *C* with vertices a_i, b_i, and c_i, i = 1,2,3, and wish to locate an observer on *A* to guard *C* while *B* blocks the visibility of *A* (see Figure 4.1). Let's refer to the guard located at vertex a_i as g_i and let z (m_i) be the height of vertex or point m_i. Suppose further;

1. $z(a_3) < z(a_1) = z(a_2)$,
2. $z(b_i) = z(a_1)$, for all *i,*
3. a_4 is the intersection point of edge a_1a_2 of triangle *A* and the line that contains edge b_1b_3 of triangle B (see Figure 4.2),
4. $x(c_1) = x(c_2) = x(c_3)$
5. By ignoring the height information, let's depict Figure 4.1 again, i.e. project it onto the x-y plane (Figure 4.2).

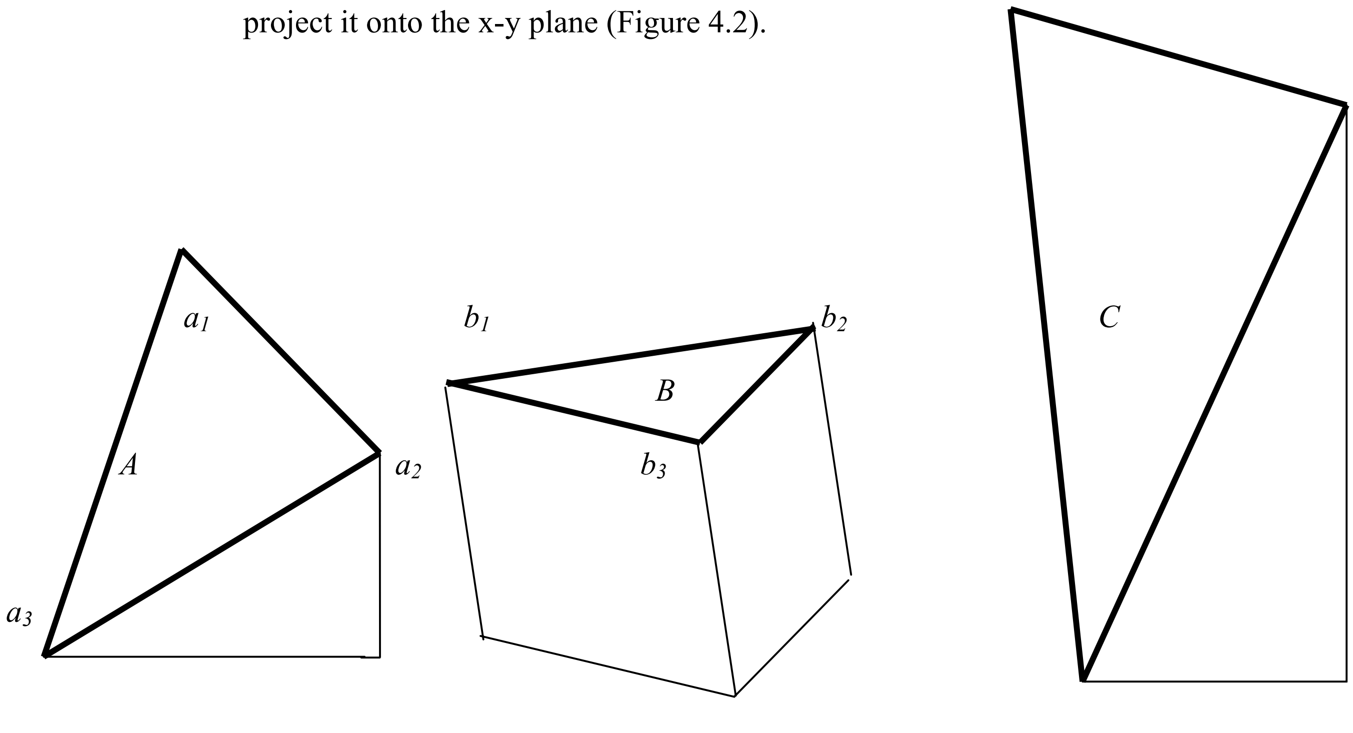


Figure 4.1.Existency of a better point guard

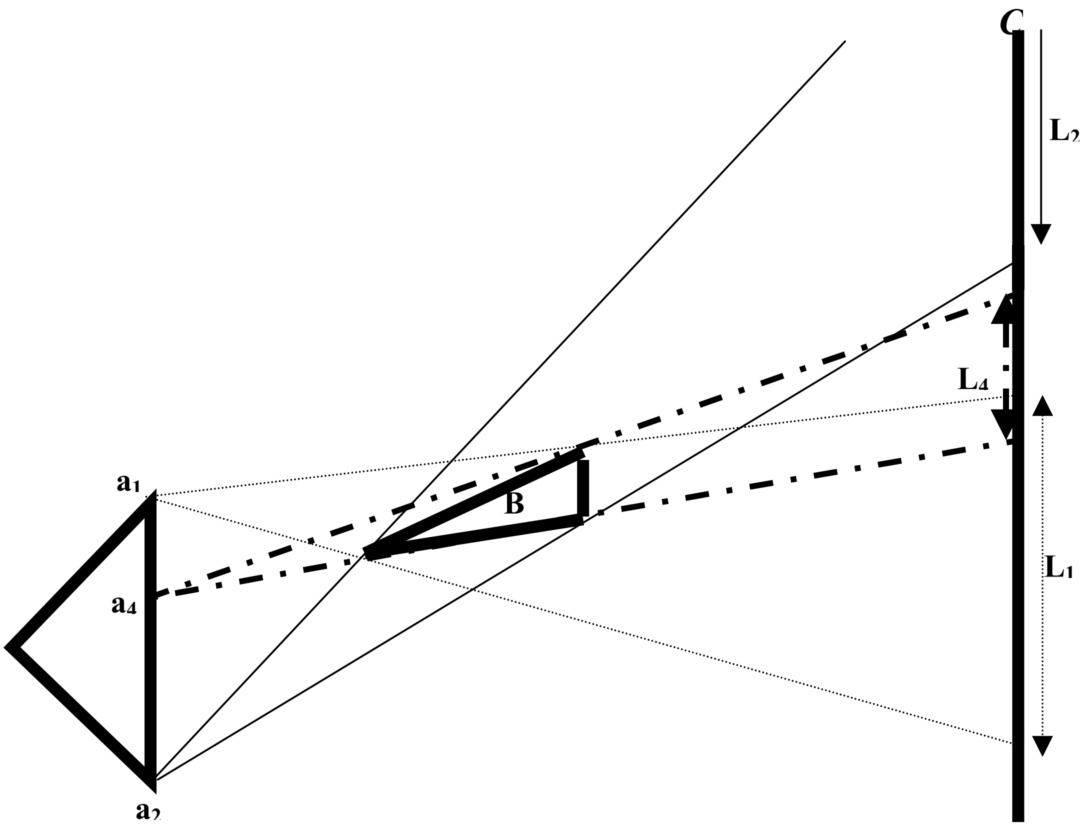


Figure 4.2: Projection of A,B, and C onto x-y plane

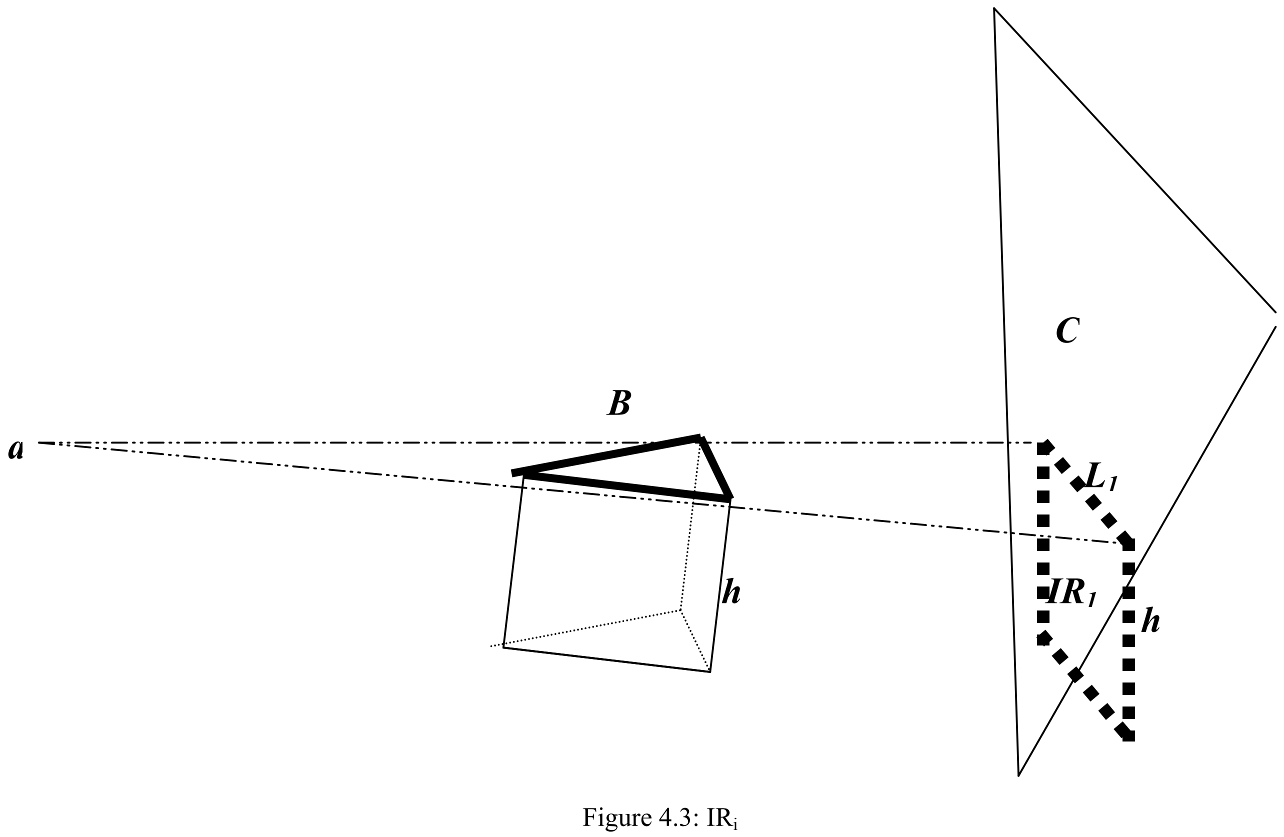


Figure 4.3: IR_i

Let IR_i be the *invisible region* of point a_i with edges e_i and f_i whose lengths are h_i and L_i, respectively. Thus IRi will be a rectangular for which h_i denotes its height and L_i denotes its length (Figure 4.3). Since A, B, and C are on the surface of a terrain and $z(a_3) < z(a_1) = z(a_2)$, g_3 can not see the region under the edge e_3 whose endpoints are points of intersection of the line segment a_1a_3 and a_2a_3 with C. Clearly, e_3 will the longest edge and h_3 will be the biggest height value.

Since g_3 is inferior, we need to find a better point guard than g_1 and g_2. Since a_1, a_2, a_4, and B have the same height, height information is trivial for the rest of the

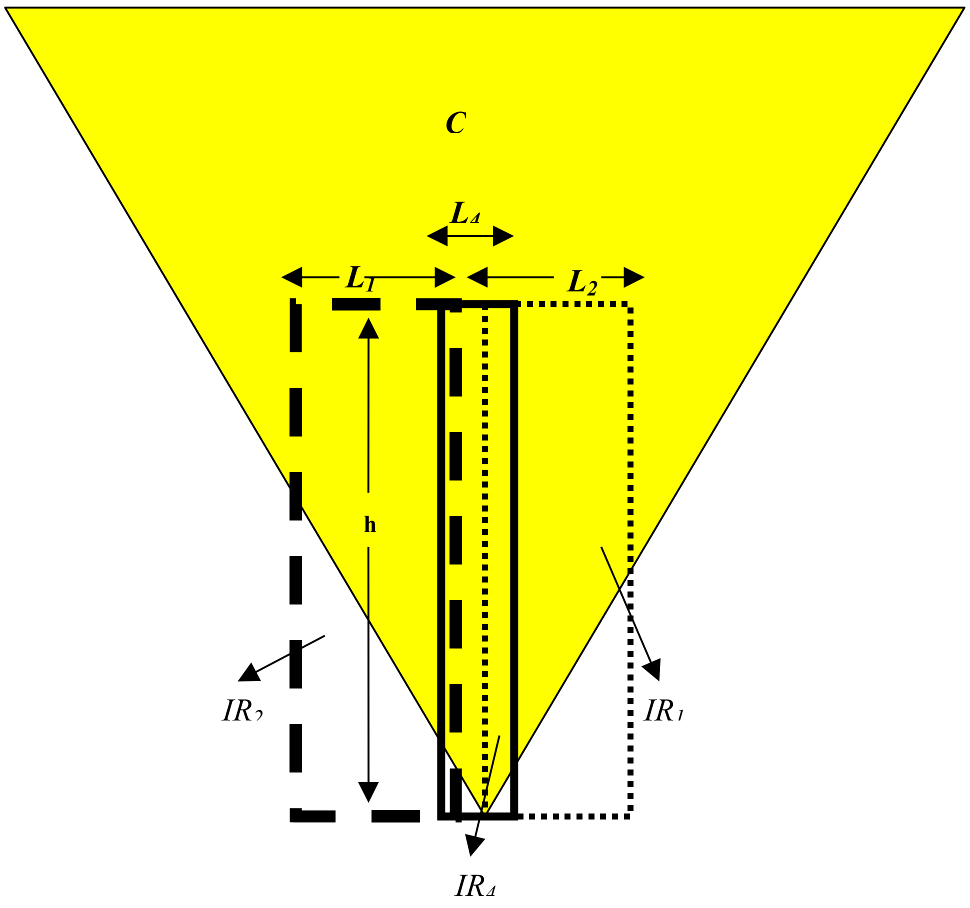


Figure 4.4: a_4 is the best location for an observer.

proof. This leads us to deal with edges of IR_i's. As shown in Figure 4.2 L_4 is the smallest and so, the invisible region with the smallest area is IR_4. Briefly,

1. $IR_3 > IR_1$ and $IR_3 > IR_2$ because $z(a_3) < z(a_1) = z(a_2)$.
2. $IR_4 < IR_1$ and $IR_4 < IR_2$ since $L_4 < L_1$ and $L_4 < L_2$ and $h_4 = h_1 = h_2$. Thus $IR_4 < IR_3$. As depicted in Figure 4.4, the point guard g_4 has a smaller invisible region than the vertex guards.

4.2 Bound for dominance

In this section, we show that a polyhedron P can be constructed for which $p_k = v_k/3$ where p_k is the number of point guards needed to guard P, and v_k is the number of vertex guards needed. To show this bound, we first construct a seven-vertex polyhedron, S_1, for which $p_k = 1$ and $v_k = 2$. By using S_1, we get a twelve-vertex polyhedron, S_2, which can be covered by three vertex guard, and conversely by two point guards. As a third step, we construct a twenty-vertex polyhedron, B_1, which is based on S_2. To cover B_1 we need six vertex guards while two point guards suffice to cover it. Finally we construct an n-vertex polyhedron, B_k, by using B_1's for which $p_k = v_k/3$.

Lemma 1. The seven-vertex polyhedral, S_1 shown in Figure 4.5 needs at least two vertex guards and if it is covered by two vertex guards, then at most one of them can be an exterior vertex. Furthermore, it can be covered by just one point guard.

Proof: Suppose that this polyhedral is a modified version of the case which we used to prove the existency of a better point guard. Namely the triangles (1) K and *A*, (2) *N, O* and B (3) *M, P* and *C* are similar. Thus vertex *x* can't see P and O because of edge *tw*, similarly *z* can't see *M* and *N* because of edge *wt.* Suppose further the height of vertex *y* is so small that the guard on vertex *y* can't see any other triangle except K, exactly like vertex a_3 in the first case. Unlike the first case, suppose the height of vertex *w* is a little smaller than the vertex *t* so that the guard located on point *q* can see vertex *t*, thus the whole seven-vertex polyhedral (Figure 4.6 shows the heights of vertices *y*, *q*, *w*, *t*).

Since vertex guards *x* and *z* can't see *O* and *N,* respectively, and vertex guards *s*, *t, u, w* can't see *K*, any of the vertex guards can not cover the whole polyhedron. Clearly vertex guards *w* and *z* (or *x*) suffice to cover the polyhedron. Let's consider the other cases;

1. If *t* is chosen, two vertex guards will also be needed to guard *M*, *P*, *L* and *K*. If one of *x* or *z* is also chosen, then two triangles will not be covered: without loss of generality, suppose vertex *x* is chosen. Then, triangles *O* and *P* will not be covered.
2. If *y* is chosen, triangles *L*, *M*, *N*, *O*, and *P* will remain unguarded. Clearly one vertex guard more will not suffice.
3. If *s* or *u* is chosen, two vertex guards more will be needed to guard *K*, and the triangles on the other side of edge *tw*. Without loss of generality, suppose s is chosen remaining *O*, *P*, *L*, and *K* uncovered. If *u* is chosen, *K* will not be covered. If *z* is chosen, *O* will not be covered.
4. If *x* or *z* is chosen, three triangles will not be covered. Without loss of generality, suppose *x* is chosen remaining *N*, *O* and *P* uncovered. If t is also

chosen, *P* will not be covered. If u is chosen *N* will remain uncovered. If *z* is chosen *O* and *N* will be uncovered.

By using this S_1, we can get a twelve-vertex polyhedral, S_2 which can be covered by three vertex guards, *x, y, z,* and conversely by two point guards, *p, q* (Figure 4.7). As the third step, we can construct a twenty-vertex polyhedral, B_1 which is based on S_2 in such a way that two S_2's are placed back-to-back (Figure 4.8). To cover S_3 we need six vertex guards while two point guards suffice to cover it since one point guard can see two back-to-back S_1's. To clarify this, it will be neccessary to find out whether one point guard can see the other S_1's vertex *t*. As shown in Figure 4.9 point *p* can see whole line between t_1 and t_2.

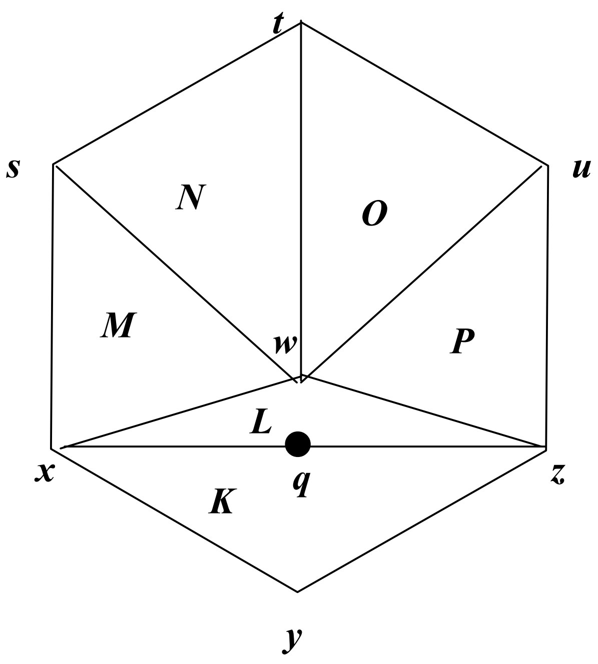


Figure 4.5: The seven vertex graph

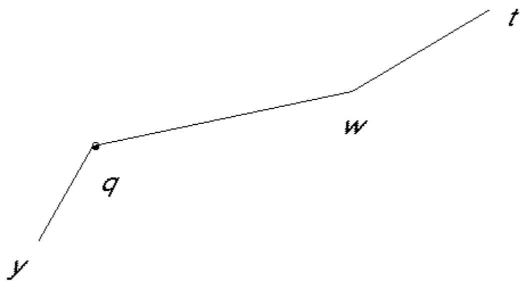


Figure 4.6: While point q can see the whole line, no vertex can.

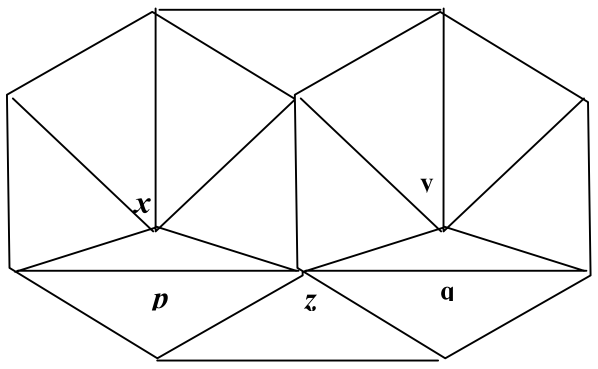


Figure 4.7: The twelve-vertex polyhedral, S_2 can be covered by three vertex guards, x, y, z, and conversely by two point guards, p, q.

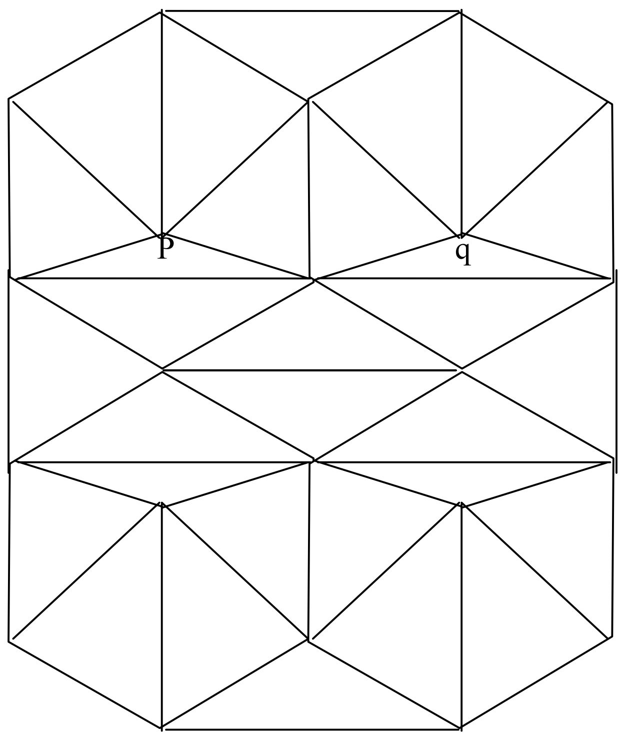


Figure 4.8: The twenty-vertex polyhedral, B_1 formed by two back-to-back S_2 , the point guards p and q suffice to cover it while it is needs six vertex guards.

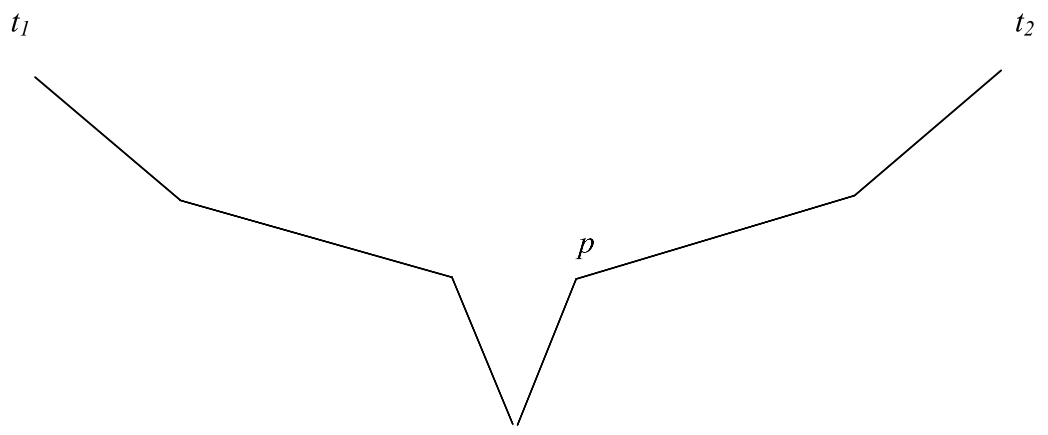


Figure 4.9: The point guard p can see the whole line

Finally we can construct an n-vertex polyhedral, B_k in a such a way that B_1's are placed side by side as shown in Figure 4.10.

Observation: B_k needs $v_k = 6k$ vertex guards and $p_k = 2k$ guards. Thus we can construct a polyhedral terrain where $p_k = v_k/3$.

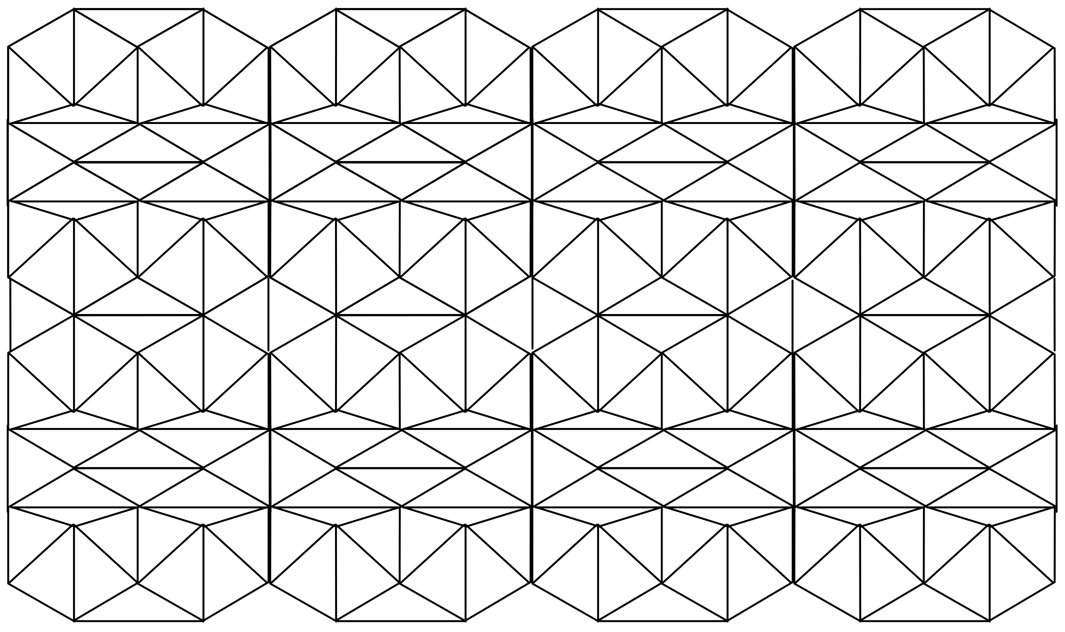


Figure 4.10: The construction of n-vertex polyhedral

4.3. Examining the dominance relationship

By using the visibility algorithm, we examined the dominance relationship in a sample terrain of approximately 5000 vertices and 10000 triangles. Among them we have examined the first 10 triangles for each of them by calculating the best vertex of the triangle and by randomly choosing 50 point guards to dominate this chosen viewpoints. As expected, just three point guards has dominated with approximately 0,6 (= 3/500) % percentage (the coding is in the appendix).

CHAPTER 5

5 Discrete Visibility

In visibility optimization problems, we have two given sets, candidate set and target set. Candidate set includes the viewpoints to be chosen to optimize the guarding and the target set includes the points to be watched. With respect to given sets we have 4 cases for visibility optimization problems depending on which of the sets are discrete or continuous. In this thesis, we have studied the first two problems.

CANDIDATE \ TARGET SET	DISCRETE	CONTINUOUS
DISCRETE	Problem 1	Problem 2
CONTINUOUS	Problem 3	Problem 4

Table 5.1: Visibility optimization problems

We have already shown in Chapter 4 that some point guards with larger areas of viewsheds than that of vertex guards can be found. This leads us to study continuous cases. Although it is possible to find precise definitions of visibility in the context of discrete models of a surface in a TIN such as the one used in this study, no such precise definition exists for continuous terrain. The results of calculating the set of vertices or points visible from each of the TIN vertices can be expressed as a rectangular matrix with each row representing a viewpoint vertex and each column a target vertex or a triangle. A target vertex is represented in each column for Problem 1 and target triangle is represented for Problem 2. Each element x_{ij} of the matrix is set to 1 if vertex i (Problem 1) or some part of triangle i (Problem 2) is visible from vertex j and 0 otherwise. Each triangle is weighted by its visible area when the objective is concerned with the area visible.

In this chapter we compute x_{ij} when both i and j are vertices, i.e. we compute the mutual visibility of two points. Problem 2 will be given in the next chapter. We compute the visible region from a vertex i iteratively, find the visible region of vertex A on triangle C and sum up the visible regions on all triangles finally.

5.1 Problem 1

The mutual visibility of two vertices A and B is determined in two ways. In the "brute force" approach, the problem is reduced to finding either the terrain edges (for a TIN), or grid cells (for an RSG) intersected by the vertical plane passing through the segment AB. For each intersected segment (edge or cell) e, a test is performed to decide whether e lies above AB, and the two points are reported as not visible in case of a positive answer for at least one of such tests (see [Lee91] for TINs and [Ray92] for RSGs). On TINs, a piecewise linear interpolation over each triangle is used in all studies. On RSGs, different tecniques are chosen like bilinear functions, or step functions, or other interpolation conventions; a common approach is to consider a linear approximation of the edges , while disregarding the interior of the cells as reported in [Nag94].

The second way can also be regarded as a special case of the ray shooting problem on a polyhedral terrain. Given a polyhedral terrain *T*, a viewpoint *V*, and a view direction (θ,α), the ray shooting problem consists of finding the first face of *T* hit by a ray emanating from *V* with direction (θ,α) (see [Col89] for details).

We also applied "brute force" approach in which two vertices of the TIN are intervisible if a straight line between them lies entirely on or above the surface without intersecting any part of the surface, except at two end points. A visibility function is usually defined as a Boolean function, $V_{AB}= 1$ when two points, *A* and *B*, are intervisible, i.e. are visible to each other. Generally a subset of triangles are defined the visible area of a vertex rather than a subset of vertices: if all vertices of a triangle *T* is visible to some vertex *V* then *T* is accepted as visible to *V*. There is no requirement that the subset be connected. However it will necessarily contain the viewpoint since we define the triangles adjacent to the viewpoint as visible.

Clearly, the visibility problem is closely related to the detection of hidden lines and hidden surface problems, which are a well-known problem in computer graphics. However, the hidden surface algorithms were designed either to determine only how an image of the surface model will appear from a given viewpoint, or for grid based surface models.

There are three main approaches when calculating the visibility of two points. The block is either;

(1) a point, generally used with regular square grid maps ([Ray92],[Fra94]),

(2) an edge of a triangle ([CS89] , [Caz91], [deF92], [GL89]) or

(3) the triangle.

To calculate the visibility of two points *A* and *B*, we will use the third approach: Try all triangles one by one to see if they block the straight line connecting the two points *A* and *B*. Let's take the case with a single triangle first. Let $T=\{v_1, v_2, v_3\}$ be the triangle with vertices v_1, v_2, v_3 where the coordinates of v_i is (x_i, y_i, z_i). Let $A=(a_1, a_2, a_3)$ and $B=(b_1, b_2, b_3)$ be the two points of interest as depicted (Figure 5.1).

5.2 The algorithm

A and *B* are intervisible if and only if the line segment connecting *A* and *B* does not intersect the triangle *T*. Whenever this is the case, some convex combination of points *A* and *B* is also a convex combination of the vertices v_1, v_2, v_3.

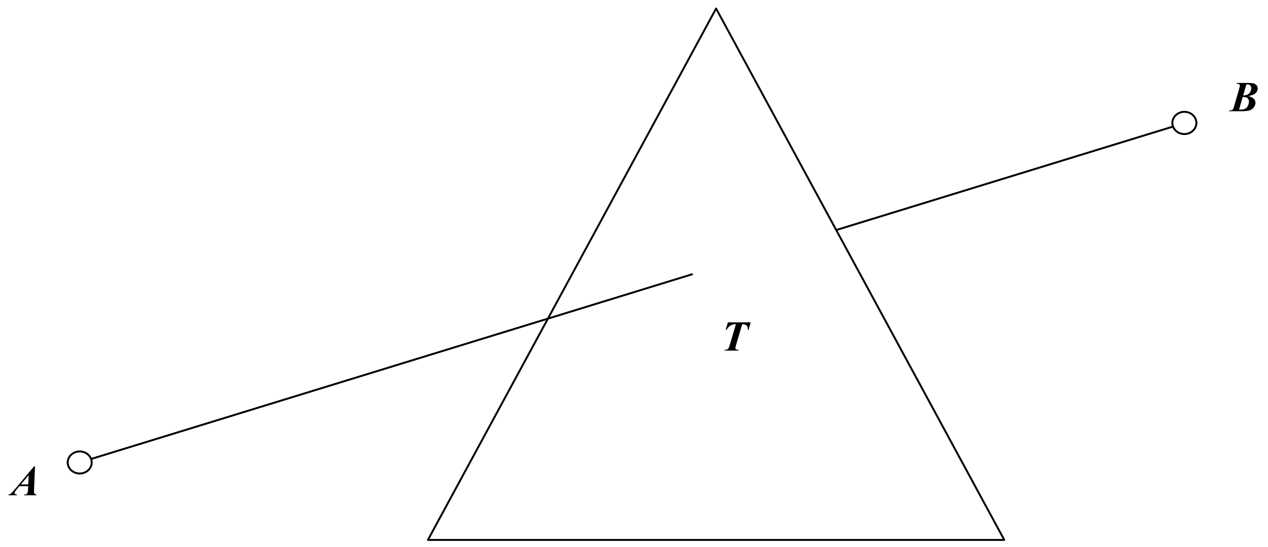


Figure 5.1: The visibility of points *A* and *B* are blocked by triangle *T*.

Thus *A* and *B* does not see each other if and only if there exists numbers $\alpha, \beta_1, \beta_2, \beta_3$ such that

$$\alpha A + (1-\alpha)B = \beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 \qquad (1)$$

with

$0 \le \alpha \le 1$,

$0 \le \beta_i \le 1$ i =1, 2, 3,

$$\beta_1 + \beta_2 + \beta_3 = 1 \;\Rightarrow\; \beta_3 = 1 - (\beta_1 + \beta_2) \qquad (2)$$

By combining (1) and (2);

$$\alpha(A-B) + B = \beta_1(v_1 - v_3) + \beta_2(v_2 - v_3) + v_3$$

which is equal to

$$\alpha(A-B) + \beta_1(v_3 - v_1) + \beta_2(v_3 - v_2) = (v_3 - B)$$

so,

$$\alpha\begin{pmatrix} a_1 - b_1 \\ a_2 - b_2 \\ a_3 - b_3 \end{pmatrix} + \beta_1\begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix} + \beta_2\begin{pmatrix} x_3 - x_2 \\ y_3 - y_2 \\ z_3 - z_2 \end{pmatrix} = \begin{pmatrix} x_3 - b_1 \\ y_3 - b_2 \\ z_3 - b_3 \end{pmatrix} \qquad (3)$$

with

$$0 \le \alpha, \beta_1, \beta_2, \beta_1 + \beta_2 \le 1 \qquad (4)$$

It follows that we are dealing with a system,(3), of 3 equations with 4 additional bounding inequalities. That is, we look for a solution to a system of the form

$Hu = t$

$0 \le u_i \le 1$, for i =1,2,3 (5)

$$0 \le u_1 + u_2 \le 1 \qquad (6)$$

Assuming H is nonsingular, Cramer's Rule gives

$$u_i = \frac{\det(H_i)}{\det(H)} \quad \text{for i =1,2,3} \qquad (7)$$

where;

$$u = \begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \end{pmatrix} \text{ and } H = \begin{pmatrix} a_1 - b_1 & x_3 - x_1 & x_3 - x_2 \\ a_2 - b_2 & y_3 - y_1 & y_3 - y_2 \\ a_3 - b_3 & z_3 - z_1 & z_3 - z_2 \end{pmatrix} \qquad (8)$$

If the solution u satisfies the bounding inequalities (4), then A and B are blocked by triangle T, i.e. they are not visible to each other. Otherwise, A and B are intervisible.

Note that H^{-1} does not exist either

1. when AB line segment is on the surface of triangle *T*, or
2. when AB line segment is parallel to the surface of *T*.

In both cases rows of H are linearly dependent. In the first case, we define A and B are intervisible as a matter of convention. If AB line segment is parallel to the surface of T, then it is enough to check the height of a point of AB line segment on *T*. For simplicity, compare the height of any vertex of T with the height of AB line segment above this vertex. If the vertex has a higher value, then they will be invisible.

5.3 Remedies for the drawbacks

Clearly there is a drawback of that method; we try all triangles one by one to see if it blocks. To overcome this, we suggest some simple operations;

1. If $v_{a\,b} = 0$ once, stop trying the other block triangles.
2. The vertices of the same triangles are visible to each other.
3. Let rectangle *R* be the rectangle whose definitive vertices are *A* and *B*. If all points of the triangle *T* are not in *R*, *T* will not clearly block the visibility of *A* and *B* (Figure 5.2).
4. If all *z* values of triangle *T* are smaller than those of both the points *A* and *B*, again *T* will not block the visibility of *A* and *B*.
5. If all vertices of T are invisible to A, then T cannot block any point C's visibility from A.

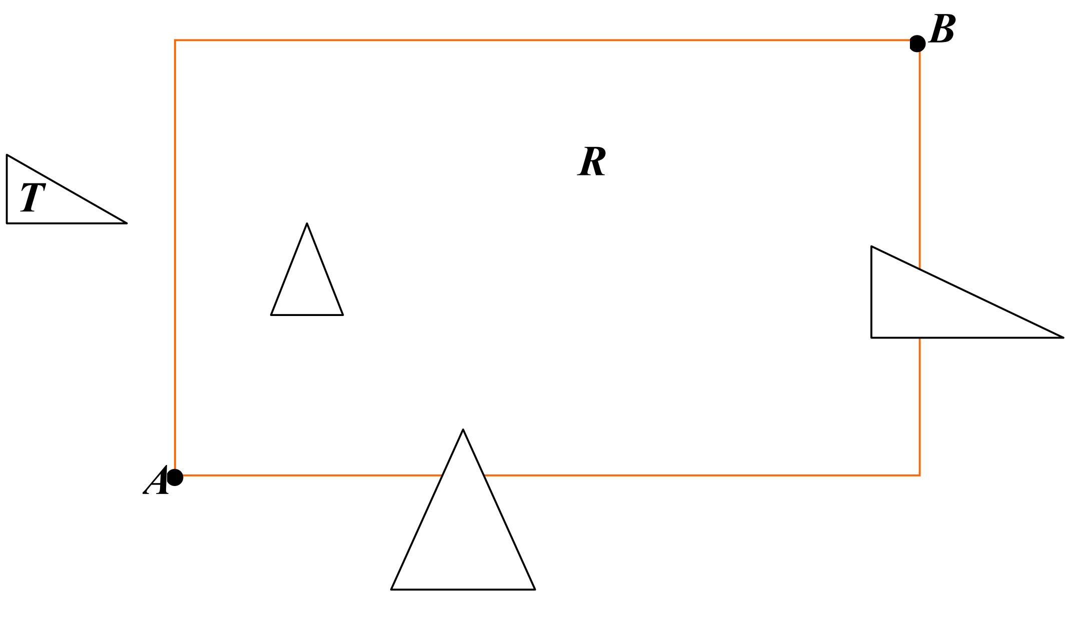


Figure 5.2: All points of *T* are outside of *R*

CHAPTER 6

6 Continuous Visibility

To calculate the visible region of a point A, we will calculate the visible region of this point on each target triangle seperately and sum up all these visible regions. Target triangle, say *C*, may be blocked completely or partially by other triangles. To find this blocked region, i.e. invisible region, it seems necessary to try all triangles other than *C*, one by one, as a blocking triangle as long as some parts of *C* is visible. Thus we will calculate the visible region of *A* on *C* in two steps;

1.) if some parts of *C* is still visible calculate the invisible region of *A* on *C* caused by one triangle, say triangle *B*, and try all triangles as *B*,
2.) find the union of these visible regions on *C*.

6.1 Step 1

Suppose we are given a point *A*, and two triangles *B* and *C*, and want to calculate the invisible region of point *A* on *C* caused by *B*. This can be done in two ways,

1.) find triangle-triangle intersection
2.) find the extreme points of the intersection region.

6.1.1 Triangle-triangle intersection

Clearly it is needed to find the projection of B on C (Figure 6.1). The projection of B on the plane containing C will also be a triangle, say S, whose vertices are the projections of the vertices of B. The vertices of S, s_i, will be on the ray $\{ A + \lambda^i d_i : \lambda^i \geq 1 \}$, where $d_i = b_i - A$ and b_i is the *i*th vertex of B. Moreover, since S and C are on the same plane, any point of S can be written as an affine combination of the vertices of C.

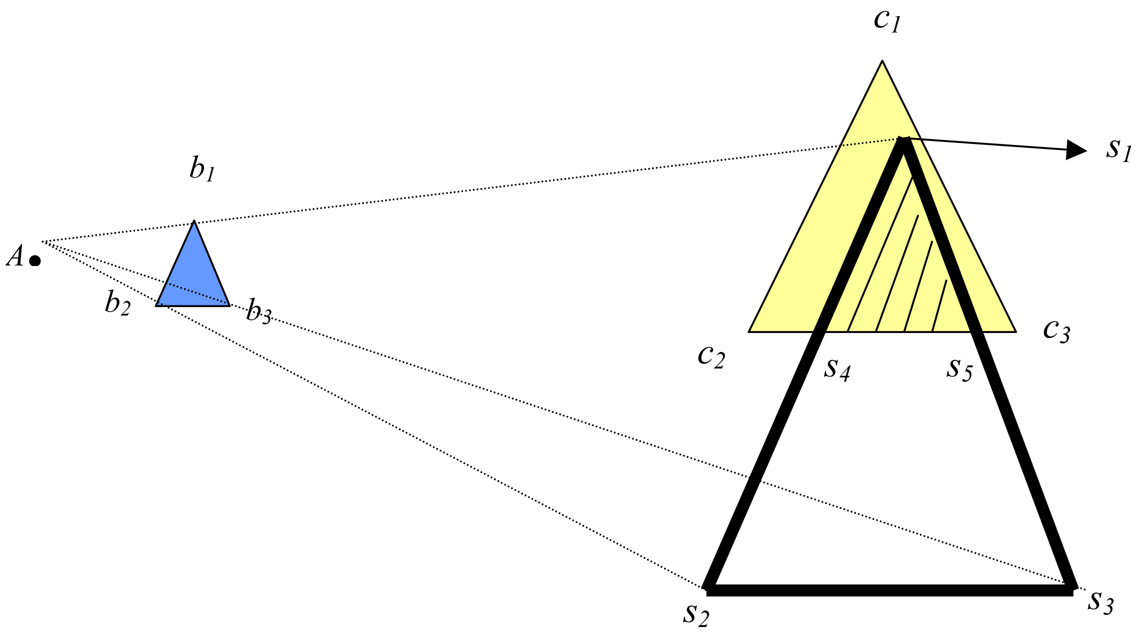


Figure 6.1: Projection of B on C

The projection of b_i (i = 1,2,3) on the plane of C requires solving the inequality system I_i below:

$$A + \lambda^i (b_i - A) = \sum_{j=1}^{3} \gamma_j^i c_j , \quad j = 1,2,3, \qquad (1)$$

$$\sum_{j=1}^{3} \gamma_j^i = 1 , \quad 0 \le \gamma^i_j \le 1 \quad j = 1,2,3, \text{ and } \lambda^i \ge 1.$$

In system I_i, we are dealing with a system of 4 equations with 4 unknowns and 4 more bounding inequalities. If system I_i has a feasible solution in the variable λ^i, γ^i_1, γ^i_2, γ^i_3 then the projection of b_i is ,

$$S_i = A + \lambda^i (b_i - A) = \sum_{j=1}^{3} \gamma_j^i c_j , \quad j = 1,2,3,$$

(1) can be rewritten as,

$$A + \lambda^i(b_i - A) = \gamma^i_1 c_1 + \gamma^i_2 c_2 + (1-\gamma^i_1-\gamma^i_2)c_3$$

$$\Rightarrow \gamma^i_1(c_1 - c_3) + \gamma^i_2(c_2 - c_3) + \lambda^i(A - b_i) = (A - c_3)$$

$$\Rightarrow (c_1 - c_3 \quad c_2 - c_3 \quad A - b_i) \begin{bmatrix} \gamma_1^i \\ \gamma_2^i \\ \lambda^i \end{bmatrix} = (A - c_3) \qquad (2)$$

(2) can be solved with either one of the two methods; Gauss-Jordan reduction and Cramer's methods. The solution gives γ^i_1, γ^i_2, and λ^i and by substituting γ^i_1, γ^i_2, and γ^i_3, which is ($1-\gamma^i_1-\gamma^i_2$), s_i will be found. In case of $\lambda^i < 1$, we conclude that B does not block C to A.

The intersection region of S and C will be the invisible region of A on C. Thus we are faced with the problem of determining the intersection region of two triangles on the same plane.

Two triangles will intersect if some convex combination of one of them is also a convex combination the other. Thus, two triangles intersect if and only if there exists α_i's and β_j's (i = 1,2,3, and j = 1,2,3) such that

$$\alpha_1 s_1 + \alpha_2 s_2 + \alpha_3 s_3 = \beta_1 c_1 + \beta_2 c_2 + \beta_3 c_3 \qquad (3)$$

with $0 \leq \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3 \leq 1$, and

$\alpha_1 + \alpha_2 + \alpha_2 = 1$,and

$\beta_1 + \beta_2 + \beta_3 = 1$

(3) can be rewritten as,

$\alpha_1 s_1 + \alpha_2 s_2 + (1-\alpha_1-\alpha_2)s_3 = \beta_1 c_1 + \beta_2 c_2 + (1-\beta_1-\beta_2)c_3$

$$\Rightarrow \alpha_1(s_1-s_3) + \alpha_2(s_2-s_3) + \beta_1(c_3-c_1) + \beta_2(c_3-c_2) = (c_3 - s_3)$$

$$\Rightarrow (s_1-s_3 \;\; s_2-s_3 \;\; c_3-c_1 \;\; c_3-c_2) \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix} = (c_3 - s_3) \qquad (4)$$

with $0 \leq \alpha_1, \alpha_2, \alpha_1+\alpha_2, \beta_1, \beta_2, \beta_1+\beta_2 \leq 1$.

It follows that we have a system of 3 equations with 4 unknowns and 6 bounding constraints. Since S and C are on the same plane, the equations are linearly dependent resulting in two free variables. Clearly we can't use Cramer's method to solve (4) instead Gauss-Jordan reduction will be used. We will demonstrate the method on the following example.

Suppose A, B, and C are given as follows, and wanted to compute the invisible region of A on C caused by B.

$$A = \begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix},\ B = \begin{bmatrix} 0.8 & 2.5 & 2 \\ 1.2 & 3 & 3.5 \\ 2 & 3 & 3 \end{bmatrix},\ \text{and } C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The projection S of B will be defined by the projection s_i of each corner b_i of B.

Thus, for i =1, we solve the system I_i;

$$\begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix} + \lambda^1 \begin{bmatrix} -2.2 \\ -3.8 \\ -6 \end{bmatrix} = \gamma_1^1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \gamma_2^1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \gamma_3^1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$ where $\gamma^1{}_3=1-(\gamma^1{}_1+\gamma^1{}_2)$. By replacing γ_3 with this,

$$\gamma_1^1 \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + \gamma_2^1 \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} + \lambda^1 \begin{bmatrix} 2.2 \\ 3.8 \\ 6 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}$$ which can be rewritten as,

$$\begin{bmatrix} 1 & 0 & 2.2 \\ 0 & 1 & 3.8 \\ -1 & -1 & 6 \end{bmatrix} \begin{bmatrix} \gamma_1^1 \\ \gamma_2^1 \\ \lambda^1 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}.$$

This system can easily be solved by Gauss-Jordan reduction. Finally we get,

$\gamma^1{}_1 = 0.25$, $\gamma^1{}_2 = 0.25$, and $\lambda^1 = 1.25$

Since, $\gamma^1{}_3 = 1 - (\gamma^1{}_1 + \gamma^1{}_2) = 0.5$.

So we have found the projection of b_1 onto the plane containing C which is;

$s_1 = (\ 0.25 \quad 0.25 \quad 0.5)^T$.

Similarly,

$s_2 = (2 \quad 1 \ \ -2)^T$ and $s_3 = (1 \quad 2 \ \ -2)^T$.

The second step is to determine whether the triangles S and C intersect or not. To decide this, we need to solve the system,

$C\alpha = S\beta$ with $\sum_i \alpha_i = 1$, $\sum_j \beta_j = 1$ and $0 \le \alpha_k$, $\beta_l \le 1$, for $k = 1,2,3$ and $l = 1,2,3$.

Thus we have,

$$\alpha_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \beta_1 \begin{bmatrix} 0.25 \\ 0.25 \\ 0.5 \end{bmatrix} + \beta_2 \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix} + \beta_3 \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}$$

with $\alpha_1+\alpha_2+\alpha_3=1$, $\beta_1+\beta_2+\beta_3 = 1$ and $0 \le \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3 \le 1$

After replacing $\alpha_3 = 1 - (\alpha_1+\alpha_2)$ and $\beta_3 = 1 - (\beta_1+\beta_2)$, Gauss-Jordan reduction is used and we get the matrix,

$$\begin{bmatrix} 1 & 0 & 0.75 & -1 & 1 \\ 0 & 1 & 1.75 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which means,

$\alpha_1 \quad + 0.75\beta_1 - \beta_2 = 1,$

$\quad \alpha_2 + 1.75\beta_1 + \beta_2 = 2.$

Equivalently,

$\alpha_1 = 1 - 0.75\beta_1 + \beta_2,$

$\alpha_2 = 2 - 1.75\beta_1 - \beta_2 .$

By using the last two equations, we get,

$0 \le \alpha_1 = 1 - 0.75\beta_1 + \beta_2 \le 1,$

$0 \le \alpha_2 = 2 - 1.75\beta_1 - \beta_2 \le 1$, and since $\alpha_3 = 1 - (\alpha_1+\alpha_2)$

$0 \le \alpha_3 = 1 - (1 - 0.75\beta_1 + \beta_2 + 2 - 1.75\beta_1 - \beta_2) \le 1,$

$\Rightarrow 0 \le \alpha_3 = -2 + 2.5\beta_1 \le 1.$

The other constraints remain the same,

$0 \leq \beta_1, \ \beta_2, \beta_3 \ \leq 1$ (5)

Finally we get,

$-1 \leq -0.75\beta_1 + \beta_2 \leq 0$, (6)

$-2 \leq -1.75\beta_1 - \ \beta_2 \leq -1$, (7)

$0.8 \leq \ \beta_1 \qquad \leq 1.2$ (8)

So, the system (5), (6), (7) and (8) give infinitely many feasible solutions as shown in the shaded region of Figure 6.2. Each choice of β1, β2 in the shaded region together with $\beta_3 = 1 - (\beta_1 + \beta_2)$ gives a point $\beta_1 c_1 + \beta_2 c_2 + (1 - \beta_1 - \beta_2) c_3$ which is in the intersection of triangles S and C.

6.1.2 Extreme points of the intersection

The second way to compute invisible region of A on C caused by B, is to find the extereme points of the intersection region. Clearly these extreme points are the edge-edge, vertex-point and point-vertex intersection points (Figure 6.3).

When the intersection point is a vertex of the projection triangle, say s_i, and a point of the target triangle, a convex combination of c_1, c_2 and c_3 (Figure 6.4), vertex-point intersection occurs. To find the intersected s_i, we use the ray information. Each such ray is an affine combination of A and one of b_j's (j = 1, 2, 3). Thus we have three systems each with three unknowns and four inequalities.

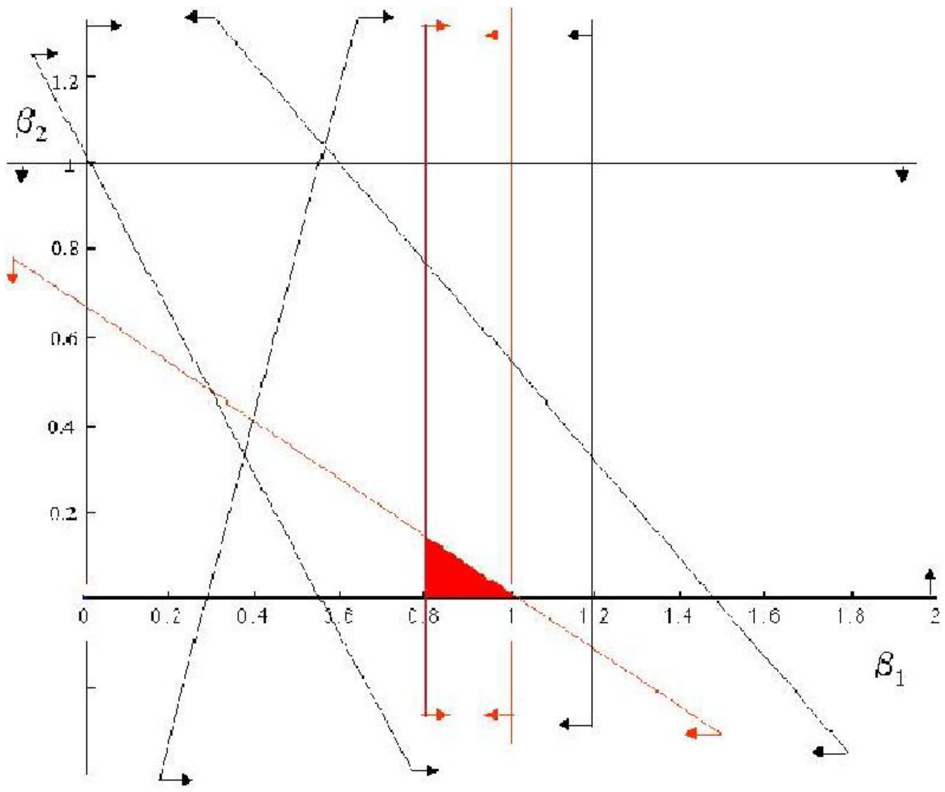


Figure 6.2. The graph of intersection .

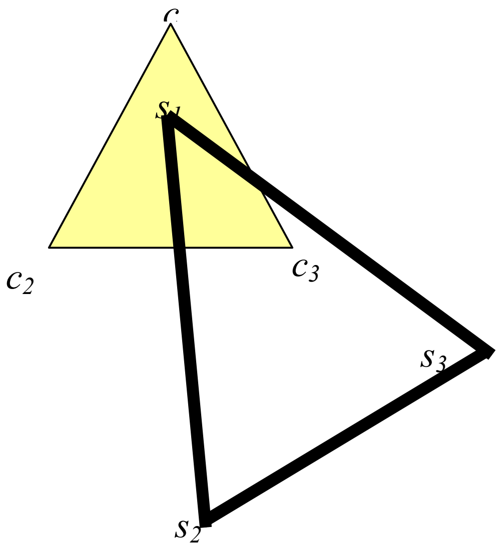


Figure 6.3: Extreme points of intersection region.

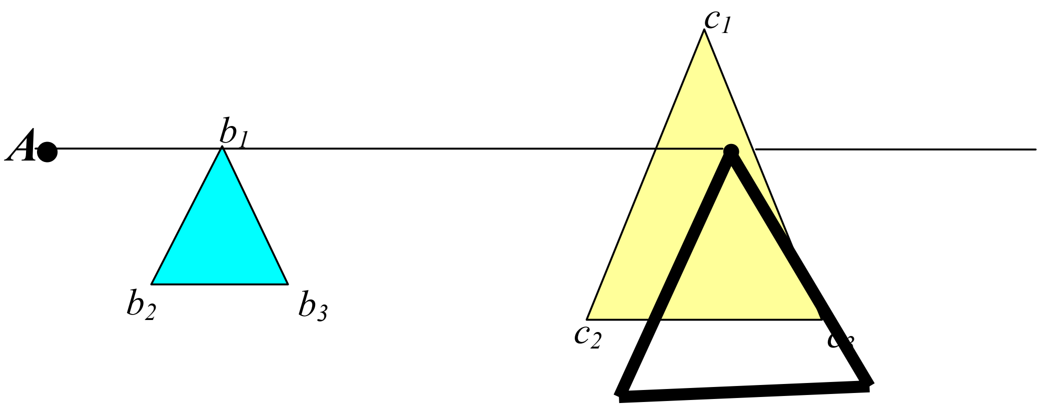


Figure 6.4: Vertex-point intersection

Observation 1: s_i intersect C if and only if there exists numbers β_1, β_2, and β_3 such that

I_i ($i = 1,2,3$)

$$s_i = \beta_1 c_1 + \beta_2 c_2 + \beta_3 c_3 \qquad (9)$$

with $0 \leq \beta_1, \beta_2, \beta_1 \leq 1$,

and $\beta_1 + \beta_2 + \beta_3 = 1.$

Since s_i is an affine combination of A and one of b_i, (9) can be rewritten as;

$$A + \alpha (b_i - A) = \beta_1 c_1 + \beta_2 c_2 + (1 - \beta_1 - \beta_2) c_3 \qquad (10)$$

With, $\alpha \geq 1$, $0 \leq \beta_1, \beta_2, \beta_1 + \beta_2 \leq 1.$

When the intersection point is the edge point of both target and projection triangles, edge-edge intersection occurs (Figure 6.5). Whenever this is the case, some convex combination of any two vertices of the projection triangle is also a convex combination of any two vertices of the target triangle.

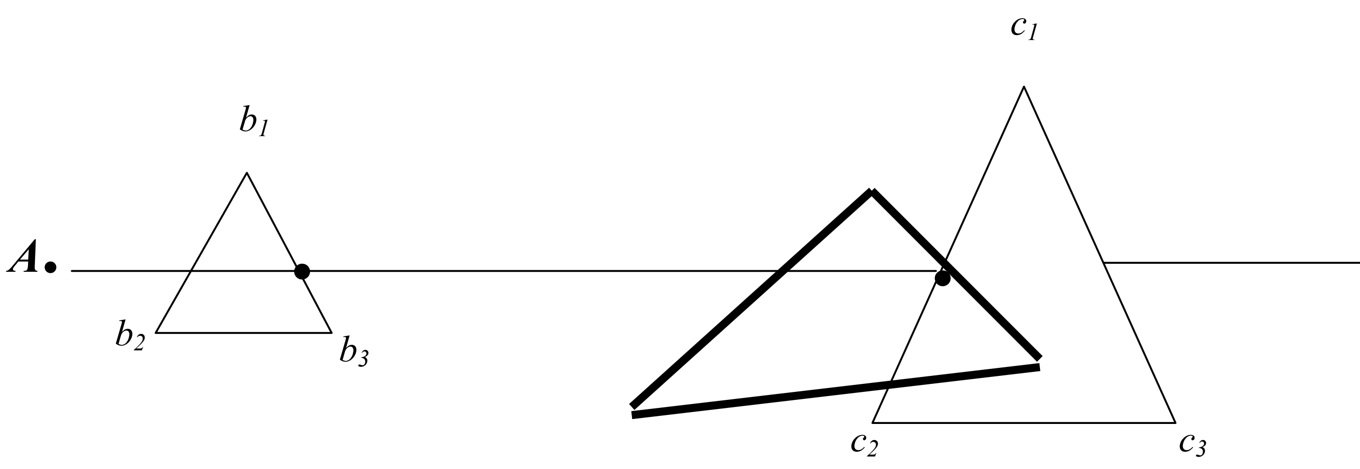


Figure 6.5: Edge-edge intersection

Observation 2: Edge s_is_j and edge c_kc_l intersect if and only if there exists numbers β and γ such that,

$\beta s_i + (1-\beta)s_j = \gamma c_k + (1-\gamma)c_l$ (11)

where; $0 \leq \beta, \gamma \leq 1$.

For each combination of $(i,j) \in \{(1,2), (1,3), (2,3)\}$ and, $(k,l) \in \{(1,2), (1,3), (2,3)\}$, we have a system of the form (11). Thus, we have nine independent systems each with three unknowns and three equalities.

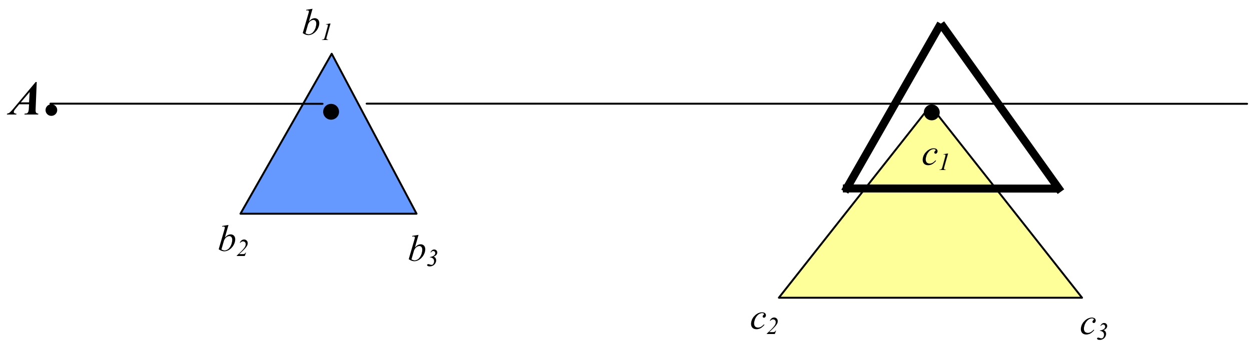


Figure 6.6: Point-vertex intersection

In the point-vertex intersection case, the point of intersection can be found in three ways where indeed all of them are the same (Figure 6.6). The first one of them is that the point of intersection is c_i and it is also a point of the of the projection triangle. That is;

$c_i = \beta_1 s_1 + \beta_2 s_2 + \beta_3 s_3$

with, $\beta_1 + \beta_2 + \beta_3 = 1$ and $0 \leq \beta_1, \beta_2, \beta_3 \leq 1$.

The second way is that the point of intersection is c_i and affine combination of the point A and some point of the block triangle. That is;

$c_i = A + \alpha(\beta_1 b_1 + \beta_2 b_2 + \beta_3 b_3)$

where $\beta_1 + \beta_2 + \beta_3 = 1$, $0 \leq \beta_1, \beta_2, \beta_3 \leq 1$ and $\alpha \geq 1$.

The last way is that the point of intersection is the convex combination of c_i and A and also it is equal to one point of the block triangle.

Observation 3: A vertex of the target triange, c_i intersect projection triangle if and only if there exist numbers α, β_1, β_2 and β_3 such that

I_i ($i = 1,2,3$)

$\alpha A + (1-\alpha)c_i = \beta_1 b_1 + \beta_2 b_2 + \beta_3 b_3$ (12)

with $0 \leq \alpha, \beta_1, \beta_2, \beta_3 \leq 1$

and $\beta_1 + \beta_2 + \beta_3 = 1$.

6.2 Step 2: The total invisible region of C

The total invisible region of target triangle to one point will be the union of invisible regions caused by different block triangles. After getting the extreme points of the union of the invisible regions, we can find this total invisible region by Monte Carlo simulation or by the algorithmic way.

6.2.1 Intersection of convex polygons

Since intersection of two convex polygons will also be convex, when we find the extreme points of this intersection, we are done. Moreover we are looking for a special case of intersections of convex polygons; an intersection of a convex polygon and a triangle. Extreme points of the intersection can be found by using (10), (11), and (12). Let e_{ij}, and e_{jk} be two edges of triangle A with vertices v_i, v_j, and v_k and B be the convex polygon (Figure 6.7). Then,

1. e_{ij} intersect B at two edges at most (because of convexity). Thus the total number of intersections cannot be bigger than six.
2. If e_{ij} intersect B at one edge, then either v_j or v_i of A intersect B (Figure 6.7). Suppose v_j does. Then v_k will also intersect B, if e_{ki} intersect B at one edge.

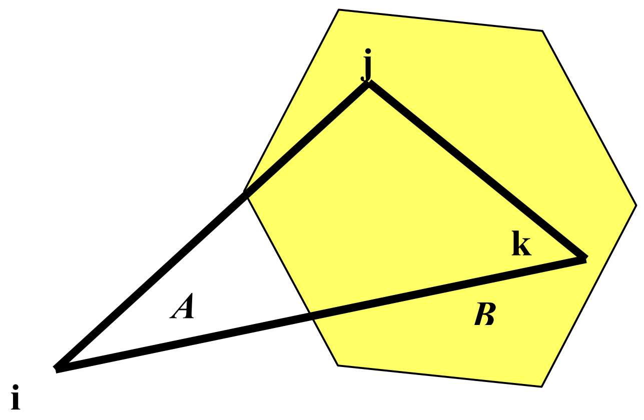


Figure 6.7:Extreme points of intersection

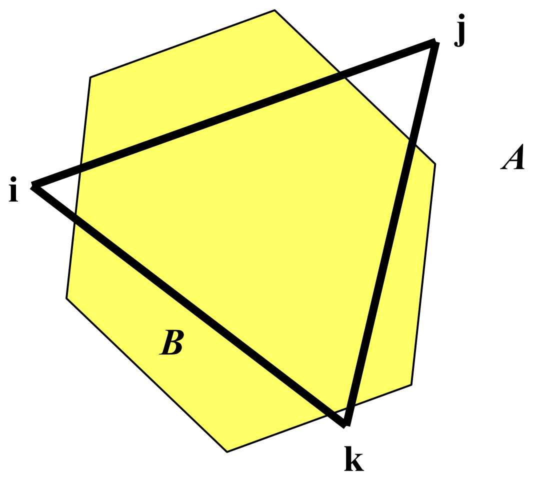


Figure 6.8: Intersection at two edges

3. If e_{ij} intersects B at two edges of B, then neither v_i nor v_j does not intersect B (Figure 6.8).
4. All of the three properties are true for also B.
5. Let $v_a, v_b, .. v_n$ be the sequence of the vertices of B and for all adjacent vertex tuples there exists an edge. If v_i and v_m intersect A and there is no edge intersection between v_i and v_m, all vertices between v_i and v_m will also intersect A.

Although for edge–edge intersections, (11) can be used, vertex intersections needs to be changed since B may have more than three vertices. In this case we can take vertices four by four and take their convex combinations. Thus we have the system;

$a_i = \beta_1 b_1 + \beta_2 b_2 + \beta_3 b_3 + \beta_4 b_4$

where $\beta_1 + \beta_2 + \beta_3 + \beta_4 = 1$ and $0 \leq \beta i \leq 1$.

So if the system

$a_i = \beta_1 b_1 + \beta_2 b_2 + \beta_3 b_3 + (1 - \beta_1 - \beta_2 - \beta_3) b_4$

where $\beta_1 + \beta_2 + \beta_3 + \beta_4 = 1$ and $0 \leq \beta i \leq 1$.

Has a solution set, then vertex intersection occurs.

Thus we can define an algorithm with two steps for finding the triangle-convex polygon intersection.

1. Find the edge-edge intersections. Remember any edge can intersect only two edges of the other. Get two lists for the edge sets of A and B, if any edge has two intersections, drop it from the list.
2. Find the vertex intersections. We can find these intersections in three steps,

(a) If any edge has two edge intersections, then it has no vertex intersection.

(b) If it has one edge intersection, it will have just one vertex intersection which needs to be calculated.

(c) As the last step, if it has no edge intersection, either both of vertices intersect or not. We can decide by using the information about the vertices of the adjacent edges. If any common vertex of the adjacent edge intersect, the other vertex of the non-intersecting edge will also intersect.

6.2.2 Monte Carlo simulation

Since we are typically dealing with thousands of triangles, this way sounds logical. Choose a point randomly on target triangle C (it should be a convex combination of the vertices of C) and check to see if this point is a convex combination of extreme points of any invisible region or not. If it is, increase the number of successes. If the process is stopped on the *n*th trial and if the number of successes is *m*, then the area of invisible region of C to A will be $a_c(m/n)$ where a_c is the area of triangle C. Change the target triangle, repeat the simulation and finally sum up all these invisible regions to find the total invisible region of A.

6.2.3 Algorithmic way

Let a_{ij} be the total area of the invisible region of a triangle *i*, to vertex *j* and a_{ijk} be the area of the invisible region of triangle *i* to vertex *j* caused by block triangle *k* (k = 1,2,...,n). Then,

$$a_{ij} = \bigcup_{k=1}^{n} a_{ijk} = \sum_{k=1}^{n} a_{ijk} - \sum_{k<l} (a_{ijk} \cap a_{ijl}) + \sum_{k<l<m} (a_{ijk} \cap a_{ijl} \cap a_{ijm}) - \ldots - (-1)^{n+1} (a_{ijk} \cap a_{ijl} \cap a_{ijm} \cap \ldots \cap a_{ijn})$$

Thus we are faced with two problems here; finding the area of a convex polygon (a_{ijk}) and finding their intersections. Since all invisible regions, not the union of any invisible region set but the projection and target triangle intersection, are clearly convex polygons, their intersection will also be convex.

It is easy to find the area of any convex polygon by first triangulating it, and then summing the triangle areas. Every convex polygon may be triangulated as a "fan" with all diagonals incident to a common vertex; and this may be done with any vertex serving as the fan "center". See Figure 6.9.

The area of a triangle is one half the base times the altitude. However this formula is not directly useful if we want the area of a triangle T whose three vertices are arbitrary points a, b, c. Let us denote this area as A(t). This base is easy: $|a-b|$, but this altitude is not so immediately available from the coordinates, unless the triangle happens to be oriented with one side parallel to one of the axes.

From linear algebra we know that the magnitude of the cross product of two vectors is the area of the parallelogram they determine: If A and B are vectors, then $|A\times B|$ is the area of the parallelogram with sides A and B. Since any triangle can be

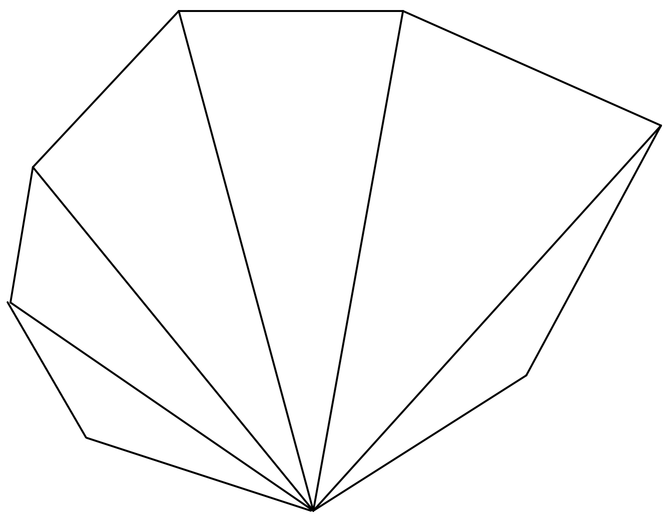


Figure 6.9: Triangulating a convex polygon

viewed as the half of a parallelogram. Just let A = b – a and B = c – a where a,b,and c are the Cartesian coordinates of the vertices of the triangle. Then the area is half the length of A×B. The cross product can be computed from the following determinant, where $\hat{i}$, $\hat{j}$, and $\hat{k}$ are unit vectors in the x,y, and z directions respectively:

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ A_0 & A_1 & A_2 \\ B_0 & B_1 & B_2 \end{vmatrix} = (A_1B_2 - A_2B_1)\hat{i} + (A_2B_0 - A_0B_2)\hat{j} + (A_0B_1 - A_1B_0)\hat{k}$$

Therefore the area of a polygon with vertices $v_0, v_1, ..., v_{n-1}$ can be calculated as

$$A(P) = A(v_0, v_1, v_2) + A(v_0, v_2, v_3) + ... + A(v_0, v_{n-2}, v_{n-1}) .$$

CHAPTER 7

7 Visibility Related Problems

Interesting application problems which are based on visibility information on a terrain, can be classified in the following major categories:

1. problems which require the placement of observation points on a topographic surface according to suitable requirements,
2. line-of-sight communication problems,
3. problems regarding the computation of paths on a terrain, with certain visibility properties.

Viewpoint placement problems require to place several observation points on a terrain, in such a way that a large part of the surface is visible from at least one point. Applications include the location of fire towers, artillery observers, radar sites, etc. In general, the aim is either to minimize the number of viewpoints to cover a target area, or, in a dual formulation, to select a fixed number of points in such a way that the visible area is maximized. By using the visibility matrix they are formulated as coverage problems. For extensive surveys on this subject see [GL89, Lee91, Nag94, Mar00].

Line-of-sight communication problems consist of finding a visibility network, connecting two or more sites, such that every two consecutive nodes of the network are mutualy visible. Applications are in the location of microwave transmitters for telephone, FM radio, television and digital data networks. A typical problem is to find the minimum number of relay towers necessary for line-of-sight communication between a set of sites. The given sites and relay towers are usually restricted on vertices of a TIN (see[Nag94, deF92, Caz91]).

Paths can be defined on a terrain, with application-dependent visibility characteristics. A *smuggler's path* is the shortest path, connecting two given points, such that no point on the path is visible from a predefined set of viewpoints. Conversely, a path where every point can be seen from all viewpoints is known as *scenic path*. Path problems are usually addressed on a DEM by restricting the viewpoints to be vertices, and the path to pass along edges. A solution (if there exists) can be determined by first computing the vertices which are visible/invisible from all viewpoints, and then applying a standard shortest path algorithm to the edges connecting them [Pup97].

In the following, we give a non-exhaustive review of relevant problems for each class, and we outline analyses and possible practical solutions of some of them.

7.1 Viewpoint placement

Problems related to the placement of observation points can be defined in terms of either a finite number of points to be watched, or of an area on the terrain to be observed. In the former case a discrete visibility model defined on $S_v \times S_o$ (respectively called the set of observation points and the set of target points)gives sufficient information for providing and testing a candidate solution to the problem.

7.1.1 Scenic sites

A typical context in which the problem of *scenic sites* (SS) arises is the detection of panoramic points along tourist paths. Such problem can be formulated (which is given in [Pup97]) as the following visibility problem:

> Given a terrain T, and two sets S_v and S_o of points on T, find the observation point p that sees as many target points as possible.

Variations of SS are *shortest watchtower* and *single observation point* problems. A shortest watchtower problem determines the location of the point with the lowest elevation above the surface from which an entire polyhedral terrain is visible (see [Sha88]). Such a point must exist because the terrain elevation is a "single-valued function", and therefore entirely visible from any point sufficiently far above it [Nag95]. The computational complexity is O($n\log^2 n$), where n is the number of polyhedral faces, so the algorithm is practicable [Sha88]. It is also possible to preprocess to determine efficiently whether any particular point is visible from a *single observation* point on or above the surface [CS89].

A solution to SS can be found by simply considering each node in S_v and by counting the number of its outgoing arcs. The worst-case time complexity is ($n_o n_v$) [Pup97].

7.1.2 Watch towers

The problem of *watch towers* (WT) relies on the surveillance of a certain area by means of a set of towers placed on the terrain: such set is required to be as small as possible, and each point within the area of interest must be seen by at least one tower. Typical application areas include the location of fire towers for forest fire monitoring, or lighthouses for safe navigation. Similar problems arise in military surveillance and weather forecasting. WT can be formulated as the following visibility problem:

> Given a terrain *T* and sets S_v and S_o of points on *T,* find a set $S' \subset S_v$ of minimum cardinality, such that any point in S_o can be seen from at least one point in S'.

The WT problem, and some of its variations, have been considered by C. Ray [Ray92], W. Franklin [Fra94], M. Marengoni [Mar00], and J. Lee [Lee91]. Variations of the problem include:

- Find the area visible from a fixed set of observation points.
- Maximize the area visible from a fixed number of observation points.
- Given some cost function related to tower height, locate the towers so as to see the entire area at minimum cost.
- Given some cost function related to tower height, locate the towers that maximize the area visible at a fixed cost.

Mainly three different heuristics have been proposed. Although no bound is warranted with respect to the optimal solution, the effectiveness of such heuristics is demonstrated by experimental results made on reasonably large data sets. The surveys and their conclusions will be given at the end of this chapter.

7.2 Line-of-sight communication

An obvious application of geometric visibility is the location of microwave transceivers for telephone, FM, and digital data networks. Of course, a realistic solution must take into account the height of the towers, the diffraction from the intermediate ridges, and the distance limit of electromagnetic propagation. So far, "only the tower-height has been considered" (claims Nagy [Nag94]). Problems involving line-of-sight communication rely on guaranteeing intervisibility between all pairs of sites of a predefined set. Relays may be permitted or not: whenever permitted, the number of relays should be as small as possible.

In the context of line-of-sight communication, using a discrete visibility model is even more strongly motivated with respect to problems related to the placement of observation points, since the predefined set of points and the set of possible relays are always finite. Furthermore, the choice for relay location is often constrained to strategic points on the terrain.

7.2.1 Communication without relays

In applications in which relays are not permitted, directional antennae are generally used at each site to ensure communication between pairs of sites. The aim is to minimize the total number of antennae. Hence, the line-of-sight communication problem consists of finding a network, whose sites are located in correspondence with a predefined set S', and connecting all points in S'. For each pair (p, q) of points of S', either q is visible from p or there exists a sequence of points connected in the network $v_1, v_2, \ldots, v_n$ in S' such that v_1 is visible from p, v_{i+1} is visible from v_i (with $1 \leq i \leq n$), and q is visible from v_n. We call such a sequence a *line-of-sight connected path* between p and q, and the global network a *connected visibility network* for S'. Directional antennae are placed at each site p in S': one antenna for each other site that is connected to p in the network. This yields the formulation of the following discrete visibility problem which is given by E. Puppo [Pup97]:

Given a terrain T, a set $S = S_v \cup S_o$ of points on T and a subset $S' \subseteq S$ find a connected visibility network for S' using the minimum number of antennae.

7.2.2 Two-point communication

The problem *two-point communication* (TPC) arises whenever two non-visible points on the terrain are given, and communication has to be warranted between them through relays. Fields of application are all those in which communication between a transmitter and a receiver station is involved (television and radio transmission, electro-magnetic signal exchange, etc.), or in which wayfinding between a departure and a destination site is concerned (navigation or terrain exploration).

Since the number of relays must be finite, a discrete visibility model is sufficient for the formulation of the problem. The following discrete visibility problem gives a formalization for TPC which can be found in [Pup97]:

Given a terrain T, a set $S = S_v \cup S_o$ of points on T and two points $p \in S_v$ and $q \in S_o$, find the minimum subset $S' \subseteq ((S_v \cap S_o) \setminus \{p, q\})$ (called the set of relays) that permits communication between p and q along a line-of-sight connected path.

Mario Cazzanti et. al. [Caz91] proposes a dynamic structure called the visibility tree to solve the TPC. They notice that the problem can be solved in two steps; first compute the visibility graph G' on S and then apply a shortest-path computation algorithm on G'. Since the computation of complete visibility graph is a quite time-consuming process, they compute point intervisibilities dynamically while computing the minimum-cost path. Therefore, the intervisibility of two points is computed only when necessary.

7.2.3 Line-of-sight network

The problem of finding a *line-of-sight network* (LSN) has relevance in several applications, such as the optimal location of radar, laser or sonar surveillance systems, of television transmitters on a predefined area on the terrain, etc. In such contexts a set S^* of transceiver/receiver stations is located on the terrain and auxiliary relays must be placed in such a way that a signal transmitted from any station can be received by any other. Clearly, the number of relays should be as small as possible. This corresponds to determine the minimum set of relays that permit connection of all stations in a connected visibility network. As already mentioned, the two-point communication problem is a special case of the line-of-sight network problem for which set S^* contains only two points. LSN can be formulated as the following discrete visibility problem (given in [deF92]):

Given a terrain *T*, a set $S = S_v \cup S_o$ of points on *T* and a subset $S^* \subset S$ (called the set of transmitter/receiver stations), find the subset $S' \subseteq ((S_v \cap S_o) \setminus S^*)$ (called the set of relays) of minimum cardinality such that points in $S \cup S^*$ can be arranged in a connected visibility network.

The above problem is equivalent to the problem known in graph theory as the *Steiner problem*, on a weighted graph in which all arcs have unit cost. The Steiner problem on graphs can be defined as follows. Given a weighted graph *G* and a subset *B* of its vertex set (called the set of the base vertices of *G*), any connected subgraph of *G* containing *B* in the set of its vertices is a Steiner subgraph of *G* for *B*. The Steiner problem in *G* with respect to *B* consists of computing a steiner subgraph of *G* for *B* of minimum cost. When the costs associated with the arcs of *G* are all positive (as in the case of visibility graph), then the Steiner subgraph of minimum cost of *G* for *B* is a Minimum Steiner Tree (MST) for *B* (see [deF92] for details).

7.2.4 Critical points

Given a set of receiver/transmission stations and a set of relays organized into a connected visibility network, the problem of *critical points* (CP) consists of determining relays that are fundamental for communication, i.e., relays whose failure would interrupt communication.Application fields in which such problem arises are those already mentioned for the LSN, in which failure detection is warranted. The problem of critical points can be formulated as the following discrete visibility problem which can be found in [Pup97]:

Given a terrain T, a set $S = S_v \cup S_o$ of points on T, a set $S^* \subset S$ of transmitter / receiver stations and a set $S' \subseteq ((S_v \cap S_o) \setminus S^*)$ of relays such that $S^* \cup S'$ form a connected visibility network, find the points whose failure disconnects the network.

7.2.5 Fault tolerant network

The problem *fault tolerant network* (FTN) combines the problem of finding a line-of-sight network with the problem of determining critical points. In several applications, such as electrical or telephonic communication, it is important to determine the minimum set of relays that allow joining a given set of transceiver/receiver stations into a connected network with the further constraint of reliability: communication among stations should be warranted even though a site fails. Failure may be due to different operational causes: overload, errors, destruction, etc. In such contexts, two antagonistic goals have to be considered: minimization of total cost and maximization of reliability. Reliability is often expressed in terms of connectivity degree of the network. While the MST connecting the given set of stations represents an optimal solution to the minimization of cost, it can be extremely vulnerable to failures. FTN can be formulated as the following discrete visibility problem [Pup97]:

Given a terrain T, a set $S = S_v \cup S_o$ of points on T and a subset $S^* \subset S$ (called the set of transmitter/receiver stations), find the subset $S' \subseteq ((S_v \cap S_o) \setminus S^*)$ (called the set of relays) of minimum cardinality such that:

- points in $S' \cup S^*$ can be arranged in a connected visibility network, and
- the failure of k points in $S' \cup S^*$ (with $1 \leq k \leq (\mid S' \cup S^* \mid - 1)$) does not disconnect the network.

7.2.6 Television broadcast

The problem *television broadcast* (TB) consists of determining the minimum number of relays necessary to connect a given transceiver station to a given set of receiver stations. A minimum number of relays may be used to this purpose. Furthermore, receiver stations cannot be employed as relays: this requirement corresponds to the practical necessity of avoiding having a receiver station located near a relay to ensure good signal reception. TB can be formulated as the following discrete visibility problem (see [Pup97]):

> Given a terrain T, a set $S = S_v \cup S_o$ of points on T, a point (called the transmitter station), and a subset $S^* \subset S_o \setminus \{p\}$ (called the set of receiver stations), find a subset $S' \subseteq (S_v \cap S_o) \setminus (S^* \cup \{p\})$ (called the set of relays) of minimum cardinality such that for each receiver station q there exists a line-of-sight connected path from p to q.

7.3 Surface paths

Finding surface paths implies the determination of lines on the terrain with specific visibility characteristics (such as scenic peculiarities, invisibility, etc). For this reasons, the solution of such problems involves retrieval of both visibility and adjacency information about points on the terrain. Since a line on the terrain is formed by an infinity of points, continuous visibility models seem more suitable in this context. However, discrete approximations can be used as well. A discrete visibility model is appropriate if, for instance, a discrete gridded model is used for terrain represention. A discrete visibility model can also be used whenever the topographic surface is represented by a TIN and lines are approximated by sequences of edges of the TIN. In both cases, reducing such problems to graph problems involves both the visibility graph and the encoding adjacency relations among points on terrain. In particular, for a gridded model, pixel-to-pixel adjacencies should be stored, while for a TIN vertex-to-vertex adjacencies should be encoded.

7.3.1 Hidden path

The problem of *hidden path* (HP) consists of determining a path on the terrain from a given site a to *a* given site *b* that is as hidden as possible from a given set of observation points. Such problem finds applications in several contexts, such as scenic landscape assessment, terrain exploration, and military topography.

Let us refer to a path on the terrain as a *topological path* to distinguish it from a *line-of-sight path*. HP can be formulated as the following discrete visibility problem ([Pup97]):

Given a set S_o of points on a terrain T, two points a and b in S_o, and a set of points S_v on T (the observation points), find a connected topological path between and having vertices in S_o such that the number of vertices of the path visible from S_v is minimum.

7.3.2 Scenic path

The problem of *scenic paths* (SP) involves the determination of a path between two given sites that is particularly interesting from a panoramic point of view. This problem can be formulated as an optimization problem, in which the number of ``boring'' points, from which no scenic sites are seen, is minimized. We can formulate SP as (see [Pup97]);

Let S_v be a set of points on a terrain T; let a, b be two points of S_v and S_o be a set of points on T. Let G_v be the visibility graph defined on $S_v \cup S_o$ and G_t be the adjacency graph defined on S. Find a connected path on G_t between a and b having vertices in a subset S' of S, and such that the number of ``boring'' nodes $p' \in S_o$ for which there exists no arc (p, p'), with $p \in S'$, is minimum.

Lee [Lee98] has surveyed some variations of surface paths which he called the *least-cost paths* including hidden path, scenic path, strategic path, and withdrawn path. Lee notices that a grid, or lattice, may be used in a GIS environment as a friction surface in least-cost path analysis, and is used to influence the computation of a least cost path on a DEM. To support the computation of the various least-cost path problems he has given, two grids of visibility information is calculated. The first is a general viewgrid (*VG*) that records for each cell the number of cells visible. The second is a dominance viewgrid (*VD*) that holds for each cell the number of cells from which the cell is visible. Two matrices are not equal because the height of the viewpoints is not ignored. Next, these two visibility grids are converted to friction surfaces. Each of four viewpaths needs different friction surfaces to be setup from viewgrids as;

- The hidden path friction surface falls between the pre-selected origin and the destination points, such that the path is minimally visible from all of the cells in the DEM. The hidden path is in fact a path that is seen the least from all parts of the terrain surface. It indicates the best route for military special forces operatios.
- The scenic path friction surface is located between the pre-selected origin and the destination point such that the path has maximum visibility of all of the cells in the DEM i.e. we wish to see the most area of the DEM surface from the path.
- The strategic path friction surface lies on the path between pre-selected origin and the destination points, and it will contain cells that have the best possibility of being minimally seen from ather cells while maintaining maximum visibility to other cells in the DEM. It indicates the best route for military surveillance and reconnaissance.
- The withdrawn path friction surface is the path between pre-selected origin and the destination points. This path is minimally visible from any cell, but it maintains minimum visibility of all of the cells in the DEM. It indicates the best route for above–ground pipelines or high-power lines.

7.4 The coverage problem

Among the visibility related problems, viewpoint placement problems are of great deal and after getting the visibility matrix, clearly they are typical of coverage problems foe which extensive surveys can be found in (see [GL89, Lee91, Nag94, Mar00]).

The results of calculating the set of triangles visible from each of the TIN vertices can be expressed as a rectangular matrix with each row representing a viewpoint vertex and each column a triangle. Each element x_{ij} of the matrix is set to 1 if the triangle *i* is visible from the vertex *j* and 0 otherwise. Each triangle can be weighted by its area if the objective is concerned with the area visible, rather than the number of triangles visible.

The formalization of the visibility coverage problems follows the standard form of of location set-covering and maximum covering location problems respectively (see [Lee91]). Let the presence of a facility at vertex *i* be denoted by y_i, which is 1 if a facility is present and 0 otherwise. To minimize the number of facilities required to see the entire surface;

$$\min \sum_i y_i$$

such that $y_i=\{0,1\}$ for all *i*

$$\sum_i x_{ij} y_i \geq 1 \quad \text{for all } j.$$

To maximize the area covered by a given number of facilities *p*, for the case of triangle *j* being accepted as visible as a whole when all vertices of it is visible to *i,* let the area of triangle *j* be denoted by A_j.

$$\max \sum_j A_j \min\left(1, \sum_i x_{ij} y_i\right)$$

such that $y_i=\{0,1\}$ for all i

$$\sum_i y_i = p \, .$$

To maximize the area covered by a given number of facilities *p*, for the case of visible part of triangle *j* is computed, replace A_j with A_{ij} where A_{ij} is the area of triangle *j* visible to *i* be denoted by A_{ij}.

Visibility dominance is used to enhance and speed up visibility analyses such as watchtower siting or viewshed assessment by Lee [Lee94]. Dominance occurs when all visible pixels from a viewing pixel are also visible from another viewing pixel. It is possible that rows can be eliminated as potential viewpoints if they are dominated by other vertices. The necessary condition for the dominance of vertex *i* by vertex *k* is simply;

$x_{kj} \geq x_{ij}$ for all j.

Unfortunately it appeares that dominance is unlikely in practice. It requires situations in which an observer is able to move on a landscape without bringing out new areas into view; in practice moves almost always result in a changing field of view, with the addition of some areas and the deletion of others. The search for coverage must therefore consider all vertices.

Three broad classes of heuristics were tested in [Lee91] also. In *greedy add* (GA) viewpoints are added to the solution set one at a time, on each step selecting the vertex which maximizes some conveniently computed parameter. To solve the problem of coverage-with-minimum observers, the GA algorithm picks for the first viewpoint the point which has the largest area of visible triangles. For the second viewpoint, the GA algorithm picks the viewpoint which has the largest area of visible triangles not visible to the first viewpoint. This process continues until the solution set just covers all the triangles in the network. Similarly for the problem of maximum-visible-area-with-n-observers the GA stops when solution set includes *n* viewpoints. The GA algorithm never removes viewpoints from the solution set. In *stingy drop* (SD) initially all vertices are selected and then SD drops the vertex with the least deterioration of the objective function. The most suitable viewpiont to be dropped is the point which produces the least areas which are visible before dropping the point, but are invisible after dropping the point. For a minimization (whole coverage) problem, the process is continued until the solution set just covers the the whole network. For a maximization (partial coverage) problem, it continues until the number of points remaining in the solution reaches the predefined number, *n*. In *greedy add with swaps* (GAS) an attempt is made to improve the objective function by exchanging each vertex in the solution with one not in the solution after the addition of each new vertex.

Two approaches had also been suggested by Lee [Lee91] to avoid very long computation time; selecting only more critical elevation vertices to reduce the size of the TINs, or selecting a subset of elevation vertices in the TIN as candidate viewpoints to reduce the size of the visibility matrices.

Tserkezou solves a small fire-tower problem, using the matrix-reduction heuristics [Tse88]. The two steps of the iterative reduction process are;

1. Eliminate any viewpoint (column) that sees only a subset of the partitions seen by some other viewpoint.
2. Eliminate any partition (row) that is seen by a superset of the viewpoints that also sees some other partition.

Marengoni et.al. [Mar00] use a 5 coloring algorithm first to reduce the number of the candidate viewpoints after simplifying the terrain by choosing fixed points of great importance and then finally use two lists to find the minimum number of observers to cover a polyhedral terrain. The first list is of observers and for each observer a list of triangles that the observer can see, as well as the total number of triangles visible to the observer. The second list gives for each triangle a list of observers that can see it. Observers are placed in the terrain using the following loop: the triangle that is viewed by the fewest observers is selected, and among those observers, the one who can see the highest number of triangles is placed in the terrain. All the triangles that the chosen observer can see are marked. The loop is repeated until all triangles are marked and the next unmarked triangle with the fewest observer is selected for the next loop.

W.R.Franklin [Fra94] proposes an algorithm to find a set of observers that jointly can see every point as follows.

1. List all the points by visibility index, and hence, to find the most visible point. Place the first observer, O_1, there.

2. Find the points that O_1 cannot see.

3. Filter the sorted list of points to delete points that O_1 can see.

4. Find the most visible point that O_1 cannot see; that is the second observer, O_2.

5. Repeat until the set of observers can see every point.

CHAPTER 8

8 Conclusion

We have shown that some point guards with larger areas of viewsheds that that of vertex guards can be found. This leads us to study continuous cases. Although it is possible to find precise definitions of visibility in the context of discrete models of a surface in a TIN such as the one used in this study, no such precise definition exists for continuous terrain; we have developed two bases for the cases of discrete viewpoints, and the cases when the viewpoints are continuous still needs to be worked.

The solution is clearly sensitive to the accuracy of the underlying TIN in representing the true topography. We have already seen that the digital elevation model can only approximate the real surface; although elevations at vertices may be exact, the surface between them is assumed to vary linearly. To obtain an accurate representation it is necessary to choose a large number of appropriately located sample points. However, an accurate TIN is no guarantee that the set of triangles visible from a point will be an accurate representation of the seen area. Apart from the artifacts such as trees which may inhibit visibility independently of the topography itself, a small error in elevation can produce very large errors in visibility while this deviation is expected to be smaller in other analyses of the same terrain such as slope or distance. For example, a difference of a few centimetres in a horizon close to the viewpoint can produce a difference of many square kilometres in the visible area. As with all location problems, the search for optimality must be conducted on a model of reality rather than on reality itself; in this case the TIN serves as a model of the real topographic surface. However the effects of the modelling are unusually explicit in this case, and the ability to investigate these effects is one of the more interesting aspects of this class of problems. It is shown that the area of the viewshed calculated in the original DEM may significantly overestimate the viewshed area [Fis91]. This study has used a Monte Carlo simulation and testing approach in which a number of randomizing models of how error occurs coded as computer procedures. However, this error was for an RSG of 200- by 200 cell and for two viewpoints. Moreover he has tested just three packages of visibility computation. Thus for TINs and other packages of visibility analyses still need to be tested whether the accuracy of visibility is significant or not.

Efficient solutions to the single viewpoint problem have been published for discrete cases of both viewpoint and candidate points in RSG and we have presented a new one for a TIN. The problem of locating the minimum number of viewpoints from which the entire terrain is visible can be solved by a set-covering algoritm. Some shortest surface-path problems constrained by visibility criteria and line-of-sight path problems are computationally tractable.

The use of the TIN digital elevation model, the restriction of the search space to the vertices of the TIN, and the definition of each vertex's visible area as a set of fully visible triangles produces a tractable version of the visibility problem which is suitable for application to site selection. It can readily be generalized to the case where viewpoints are raised above the surface, and standard heuristics for set covering are presented and still needs to be worked.

Still many visibility-related problems on terrains lack of practically satisfactory solutions. This lack of efficient solutions is partially due to the fact that such problems have deserved only little attention from the research community, both because most of them are intrinsically hard , and because some of them have come to the attention of Geographic Information Systems (GIS) people only recently . The major driving force has been geographic information systems for military applications. The applications of visibility methods for navigation for civilian purposes has barely been initiated. In many cases a good definition of the problem is still missing, thus making the work of finding algorithmic methods even harder.

References

[Aga94] Agarwal, P.K. Surface approximation and geometric partitions, *Proceedings 5th ACM-SIAM Symposium on Discrete Algorithms,* pp. 24-44, (1994).

[Aro89] Aronoff, S., *Geographic Information Systems: A Management Perspective*, WDL Publications, Ottawa, (1989).

[AT81] Avis, T. and Toussaint, G.T., An efficient algorithm for for decomposing a polygon into star-shaped polygons. Pattern Recognition, vol.13-6, pp. 395-398, (1981).

[Bos97] Bose, P. Guarding polyhedral terrains. *Computational Geometry: Theory and Approximations,* vol. 3. February, pp.173-1.86, (1997).

[Bur86] Burrough, P.A., *Principles of Geographic Information Systems for Land Resources Assessment*, Clarendon Press, Oxford, (1986).

[Caz91] Cazzanti, *M.* Visibility computation on a TIN. *Progress in Image Analysis and Processing II,* Villa Olmo, Como,Italy, 4-6 September pp.721-728 (1991) .

[Chv75] Chvatál, V. A combinatorial theorem in plane geometry. *Journal of Symbolic Computation*, vol. 7, pp.11-30, (1989).

[CS89] Cole, R. and Sharir, M. Visibility problems for polyhedral terrains. *J. Symbolic Computation,* no 7, pp. 11-30 (1989).

[deB97] de Berg, M. Computational Geometry. Springer,1997.

[dFP$^+$86] de Floriani, L., Falcidiano, Pienovi, Allen, and Nagy, A visibility based model for terrain features. *Proc. 2nd International Symposium on Spatial Data Handling*, pp. 235-250, (1986).

[deF92] de Floriani, L. Computing line-of-sight network on a terrain model. *Proceedings 5th Int.Symposium on Spatial Data Handling*, August 3-7,Charleston, South Carolina. USA, Vol.2 pp.672-681 (1992).

[dP92] de Floriani, L. and Puppo, E. A hierarchical triangle-based model for terrain description. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space,* pp. 236-251, (1992).

[DLR90] Dyn, N., Levin, D. and Rippa, S.. Data dependent triangulations for piecewise linear interpolation. *IMA J. Numer. Anal.,* 10(1), pp.137-154, Jan. (1990).

[Fis78] Fisk, S. A short proof of Chvatál's watchman theorem. *Journal of Combinatorial Theory* , ser B 24, pp.374, (1978).

[Fis91] Fisher, P.F. First experiments in viewshed uncertainity: the accuracy of the viewshed area. *American Society For Photogrammetric Engineering and Remote Sensing,*Vol.57, no10, october pp.1321-1327 (1991).

[For87] Fortune, S. A sweep line algorithm for Voronoi diagrams, *Algorithmica,* vol. 2(2), pp. 153-174, (1987).

[Fra94] Franklin, W.R. Higher is not necessarily better: visibility algorithms and experiments. Tech. Rep. Rensselaer Polytechnic Institute (1994).

[GL89] Goodchild, M.F. and Lee, J. Coverage problems on visibility regions on topographic surfaces. *Annals of Operations Research,* vol 18, pp. 175-186, (1989).

[HG95] Heckbert, P.S.and Garland, M. Survey of surface approximation algorithms. Technical report, CS Dept., Carnegie Mellon U., (1995). CMU-CS-95-194, URL = http://www.cs.cmu.edu/~garland/scape.

[LS90] Lee, D.T. and Schacter, B.J. Two algorithms for constructing Delaunay triangulations*, International Journal of Computer and Information Systems,* no.9 (3), pp. 219-242, (1980) .

[Lee91] Lee, J. Analyses of visibility sites on topographic surfaces. *Int. J. Geographical Information Systems,* Vol. 5, No. 4, pp.413-429 (1991).

[Lee94] Lee, J. Digital analysis of viewshed inclusion and topographic features on digital elevation models*, Photogrammettic Engineering & Remote Sensing,* vol. 60, no. 4, pp. 451-456, April (1994).

[Lee98] Lee, J. On applying viewshed analysis for determining least-cost paths on Digital Elavation Models. *Int. J. Geographical Information Science,* vol. 12, no. 8, pp. 891-905, (1998).

[Mar00] Marengoni, M. Placing observers to cover a polyhedral terrain in polynomial time. *Vision And Image Computing,* Vol.18, No.10, pp.773-780 (2000).

[Nag94] Nagy, G. Terrain visibility. *Comp.&Graphics*, Vol. 18, No. 6, pp. 763-773 (1994).

[Nag95] Nagy, G. Geometry and Geographical Information Systems. *Geometry At Work,* no 53, pp.88-104 (1995).

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

[O'R98] O'Rourke, J. *Computational Geometry in C*. Cambridge University Press,(1998).

[Pet90] Petrie, G. *Modelling, interpolation and contouring procedures, in Terrain Modelling in Survey and Civil Engineering*, Whittles Publishing-Thomas Telford, London, pp. 112-127, (1990).

[PS85] Preparata, F.P., Shamos, M.I. *Computational Geomatry: an introduction*, Springer verlag, (1985)

[Pup97] Puppo, E. Discrete visibility problems and graph algorihms. *International Journal of Geographical Information Systems,* 11, 2, pp.139-162, (1997).

[Ray92] Ray, C.Geometric algorithms for siting air defence missile batteries (TR 2756). Rensselaer Polytechnic Institute (1992)

[Sha88] Sharir, M. The shortest watchtower and related problems for polyhedral terrains. *Information Processing Letters,* no 29 pp. 265-270 (1988)

[She92] Shermer, T. Recent results in art galleries. *Proc. IEEE, Special Issue on Comp. Geom., September*, (1992).

[Tse88] Tserkezou, P. Characteristics of visibility models. MS thesis, Rensselaer Polytechnic Institute, Troy, NY, (1988) [Web90] Webb, H. Creation of digital terrain models using analytical photogrammetry and their use in civil engineering. *Whittles Publishing-Thomas Telford, London,* pp. 73-84, (1990)

APPENDIX

CODE FOR CHECKING DOMINANCE

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>


// 5329 Verticies
// 10368 Triangles

static short face_indicies[10368][9] = {
// QuadPatch0
        {1729,1730,1735 ,0,1,2 ,0,1,2 }, {1735,1734,1729 ,2,3,0 ,2,3,0 },
        {1730,1731,1736 ,1,4,5 ,1,4,5 }, {1736,1735,1730 ,5,2,1 ,5,2,1 },
*
*
*
,5296,5287,5282 ,5302,5296,5294 }
};
static float vertices [5329][3] = {
{-0.499842f,-0.499826f,-0.0133705f},{-0.416546f,-0.499826f,-
0.015398f},{0.333251f,-0.499826f,-0.00900962f},
*
*
*
{0.485809f,0.48584f,-0.0141254f}
};


int visibility_control(float [],int ,int);
float det(float [3][3]);
int boundry_control(float [],float ,float ,float ,float [3][3]);


void main()
{

  int N_VERTICES, N_TRIANGLES;
  int visibility[6000][6000];
  int ij,i,j,k;
  float base[3];
```

```
   int v_number[3];
   int vis_vertex[3];
   int viewpoint[10],vis_triangle[10];
   float v[3][3],v4[3];
   int alfa[3];
   int visibility2[5000];
   int vis_inner[50];
   float success_point[50][3],success_percentage;
   int success;


   N_VERTICES=sizeof(vertices);
   N_TRIANGLES=sizeof(face_indicies);


 //make all the elements visibility matrix 2 which means it is not tried
  for (ij=0;ij<9;ij++)
  {
  for (i=0;i<N_VERTICES;i++)
      for (j=0;i<N_VERTICES;i++)
          {
          visibility[i][j]=2;
          visibility[j][i]=2;
          }
 //vertices in the same triangles are visible to each other
  for (i=0;i<N_TRIANGLES;i++)
     { for (j=0;j<3;j++)
          for (k=j;k<3;k++)
            {
            visibility[face_indicies[i][j]][face_indicies[i][k]]=1;
            visibility[face_indicies[i][k]][face_indicies[i][j]]=1;
            }
     }


  // take the ijth Triangle as the Base Triangle

  for (i=0;i<3;i++)
      {
       v_number[i]=face_indicies[ij][i];//the number ith vertex of the base triangle
       for (j=0;j<3;j++)
       base[j] = vertices[v_number[i]][j]; //the coordinates of the ith vertex of the
base triangle
```

```
          //can base see vertex j, try all vertices other than the invisible vertices
          //and the vertices which are in the same triangle as vertex base

         for (int j=0;j<=N_VERTICES;j++)
           {
             if (visibility[v_number[i]][j]==2)   //if it is not tried
               {
                 visibility[v_number[i]][j]=visibility_control(base,j,N_TRIANGLES);
                 visibility[j][v_number[i]]=visibility[v_number[i]][j];
                 vis_vertex[i]=vis_vertex[i]+visibility[v_number[i]][j];
                 }
             }
        }

  //choose the best vertex of the triangle
  if ((vis_vertex[0] > vis_vertex[1]) && (vis_vertex[0] > vis_vertex[2]))
           {
           viewpoint[ij]=v_number[0];
           vis_triangle[ij]=vis_vertex[0];
           }
  else
     {if ( vis_vertex[1] > vis_vertex[2])
           {
           viewpoint[ij]=v_number[1];
           vis_triangle[ij]=vis_vertex[1];
           }
      else
         {
         viewpoint[ij]=v_number[2];
         vis_triangle[ij]=vis_vertex[2];
         }
     }

 //take an inner point and check if its viewshed is larger than that of the triangle
 for (i=0;i<50;i++)
   {
     for (j=0;j<3;j++)
        {
         for (k=0;k<3;k++)
           v[j][k]=vertices[v_number[j]][k];
        }

     alfa[0]=1;
     alfa[1]=1;
     alfa[2]=1;
```

```
       if (((alfa[0]+ alfa[1])>1)||(alfa[0]<0)||(alfa[1]<0)) {
             for (j=0;j<2;j++)
                 alfa[j]=rand(); }

    alfa[2]=1-(alfa[0]+alfa[1]);

    for (j=0;j<3;j++)
        v4[j]=(alfa[0]*v[0][j])+(alfa[1]*v[1][j])+(alfa[2]*v[2][j]);

    //create a visibility index vector, of whose all values are 2, for the current inner
point
     for (j=0;j<N_VERTICES;j++)
       visibility2[j]=2;
     //the vertices of the base triangle are visible to the inner point
     for (j=0;j<3;j++)
        visibility2[v_number[j]]=1;

    //find whether viewshed of this inner point is better than the chosen viewpoint
    for (int j=0;j<=N_VERTICES;j++)
             {
               if (visibility2[j]==2)   //if it is not tried
                  {
                   visibility2[j]=visibility_control(v4,j, N_TRIANGLES);
                   vis_inner[i]=vis_inner[i]+visibility2[j];
                  }
              }

   if(vis_inner[i]>vis_triangle[0])
      {
       for (i=0;i<3;i++)
       //success_point[i]= v4[i];
       //success_percentage=((vis_inner[i])/(vis_triangle[ij])) * 100;
       success++;
      }
   }
}
cout << success << endl ;
}

//**************FUNCTIONS*****************

  //are vertices A and B  blocked by triangle T

int visibility_control(float a[],int b,int N_TRIANGLES)
```

```
{
 float bx,by,bz,x[3][3],H[3][3];
 float NH[3][3],t[3],u[3], a1 [3];
 int conc,i,j,k,b_control;
 float deth,dethi;

  bx=vertices[b][0];
  by=vertices[b][1];
  bz=vertices[b][2];
  conc=1;

  for ( i=0 ;i<3;i++ )
                                        a1 [i] = a [i] ;

 for (k=0; N_TRIANGLES;k++)
    {
     if (conc==1)
       {
        for (i=0;i<3;i++)
          for (j=0;j<3;j++)
              x[i][j]=vertices[face_indicies[k][i]][j];
            b_control=boundry_control(a1,bx,by,bz,x)  ;
         if (b_control==1)
         {
          for (i=0;i<3;i++)
              H[i][0] = a[i]-bx;
          for (i=0;i<3;i++)
              H[i][1] = x[2][i]-x[0][i];
          for (i=0;i<3;i++)
              H[i][2] = x[2][i]-x[1][i];
          for (i=0;i<3;i++)
              t[i]=x[2][i]-bx;

         deth=det(H);

         for (i=0;i<3;i++)
             {
             for (j=0;j<3;j++)
                for (k=0;k<3;k++)
                    NH[j][k]=H[j][k];
             for (j=0;j<3;j++)
                NH[j][i]=t[j];
             dethi=det(NH);
             u[i]=dethi/deth;
             }
if(u[0]<=1&&u[0]>=0&&u[1]<=1&&u[1]>=0&&u[2]<=1&&u[2]>=0&&u[0]+u[1]
+u[2]==1)
   conc=0;
        }
      } // end of outer if
    } // end of outer for

   return conc;
  }

  //determinant

float det(float h[3][3])
  {
  float conclusion;

   conclusion=
h[0][0]*h[1][1]*h[2][2]+h[0][1]*h[1][2]*h[2][0]+h[0][2]*h[1][0]*h[2][1]
          -h[0][1]*h[1][0]*h[2][2]-h[0][0]*h[1][2]*h[2][1]-h[0][1]*h[1][1]*h[2][0]  ;

   return conclusion ;
  }

int boundry_control(float a[],float bx,float by,float bz,float x[3][3])
  {
   float upperx,uppery,lowerx,lowery;
   int control,i;

    if (a[0]<=bx)
      {
       upperx=bx;
       lowerx=a[0];
      }
     else
      {
       upperx=a[0];
       lowerx=bx;
      }
    if (a[1]<=by)
      {
       uppery=by;
       lowery=a[1];
      }
     else
  {
    uppery=a[1];
    lowery=by;
  }

 for ( i=0;i<3;i++)
 if ((x[i][0]<= upperx) && (x[i][1]<= uppery)
     && (x[i][0]>= lowerx) && (x[i][1]>= lowery))
        control=1;

 for ( i=0;i<3;i++)
      if(x[i][2]<=a[2]&&x[i][2]<=bz)
        control=0;
 return control ;
}
```