Computation and complexity of visibility in geometric environments

Computation and complexity of visibility in geometric environments

Berekenen en complexiteit van zichtbaarheid in geometrische omgevingen
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. J.C. Stoof, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 9 april 2008 des middags te 12.45 uur

door

Esther Jantje Moet

geboren op 28 februari 1980 te Dirksland

Promotor: Prof.dr. M. H. Overmars

Co-promotor: Dr. M. J. van Kreveld

Contents

1 Introduction **1**
1.1 Motivation . . . 2
1.1.1 Visibility in robotics . . . 2
1.1.2 Visibility in GIS . . . 3
1.1.3 Visibility in computer graphics . . . 4
1.2 Visibility problems in computational geometry . . . 5
1.2.1 Modeling the world . . . 6
1.2.2 Modeling visibility . . . 7
1.3 Types of problems . . . 9
1.3.1 Guarding problems . . . 10
1.3.2 Computing what is visible . . . 11
1.3.3 Analytical data structures . . . 12
1.4 Contributions of this thesis . . . 13

2 Guarding art galleries by guarding witnesses **15**
2.1 Introduction . . . 16
2.2 Preliminaries . . . 17
2.3 Examples . . . 21
2.4 Finite witness sets . . . 23
2.5 Minimum size witness sets . . . 28
2.5.1 Reflex-reflex and convex-reflex edges . . . 28
2.5.2 Convex-convex edges . . . 29
2.5.3 Placement of witnesses . . . 29
2.5.4 Size of a minimum size witness set . . . 30
2.6 Algorithms . . . 31
2.6.1 Visibility kernel arrangement . . . 31
2.6.2 Algorithms . . . 33
2.7 Concluding remarks . . . 35

3 Region intervisibility in terrains **37**
3.1 Introduction . . . 38
3.2 Preliminaries . . . 39
3.3 Complete intervisibility . . . 40

3.3.1 Geometric properties . . . 40
3.3.2 Algorithm . . . 42
3.3.3 Running time analysis . . . 44
3.3.4 3SUM-hardness . . . 45
3.4 Semi-complete intervisibility . . . 47
3.4.1 Aspect graph approach . . . 48
3.4.2 Using the edge visibility map . . . 49
3.5 Partial intervisibility . . . 50
3.6 Concluding remarks . . . 51

4 Visibility maps of segments and triangles in 3D 53
4.1 Introduction . . . 54
4.2 Preliminaries . . . 56
4.3 The strong visibility map of an edge . . . 58
4.4 The weak visibility map of an edge . . . 63
4.4.1 Lower bounds . . . 63
4.4.2 Upper bounds . . . 64
4.4.3 Algorithm . . . 66
4.5 The triangle visibility map . . . 68
4.5.1 The strong visibility map of a triangle . . . 68
4.5.2 The weak visibility map of a triangle . . . 69
4.5.3 Algorithm . . . 71
4.6 Concluding remarks . . . 72

5 On realistic terrains 73
5.1 Introduction . . . 74
5.2 Input model . . . 76
5.2.1 No subset of assumptions is sufficient . . . 79
5.2.2 Auxiliary results . . . 82
5.2.3 Dihedral angles . . . 84
5.3 The visibility map . . . 86
5.3.1 Upper bounds . . . 86
5.3.2 Lower bounds . . . 88
5.4 The shortest path map and the Voronoi diagram . . . 90
5.4.1 Bisectors . . . 90
5.4.2 Shortest path maps . . . 92
5.4.3 Voronoi diagram . . . 97
5.5 Concluding remarks . . . 99

6 Experimental verification of a realistic input model for terrains 101
6.1 Introduction . . . 102
6.2 Experimental setup . . . 103
6.2.1 DEM data . . . 103
6.2.2 TIN generation . . . 104
6.3 The model . . . 108

6.4 Evaluation of the model parameters 108
6.4.1 Aspect ratio 108
6.4.2 Fatness 109
6.4.3 Edge length ratio 111
6.4.4 Dihedral angle 112
6.4.5 Computation 113
6.4.6 Results 114
6.5 Conclusions 117
Screenshots of the terrains 118

7 Conclusions **127**

Acknowledgements **132**

Bibliography **133**

Samenvatting **147**

Index **146**

Curriculum vitae **151**

Chapter 1

Introduction

For many research fields, it is clear what you study and why. In medicine, we want to make people as healthy as possible. In biology and physics, we try to understand the physical world around us. In history, we study the past to learn from our mistakes for the future. But for many people, it is not very clear what a computer scientist actually does. One way to describe our daily job is that we try to describe the world around us mathematically, and then solve problems from this real world with the aid of a computer. Problems that are studied by computer scientists usually originate in other areas of research or in applications in business or industry. These problems come to the attention of computer scientists because other people encountered these problems in real life, and were not able to solve them with the techniques that they had at their disposal, or not within a reasonable time frame, or not without a computer.

To be able to perform a (theoretical) computer science study, it is necessary to abstract from the applications and concentrate on the fundamental mathematical problems that lie underneath. In this thesis, we study visibility problems, which are geometric in nature. This leads us to the field of computational geometry (CG). The main topic of this thesis is visibility amidst objects in two and three dimensional environments.

Studying fundamental and theoretical problems does not mean we can forget about the motivation behind what we do. Even the reason for studying a particular fundamental problem can still come from a real-world application. When we study abstract problems, whose solutions sometimes are merely of theoretical interest, it is important to keep in mind exactly why we study these problems. Therefore, we give an extensive overview of applications and application areas in which visibility plays a role. This motivation covers a large part of this first chapter. After that, we discuss visibility research in computational geometry: previous research and the topics of this thesis, and the relation between these two.

1.1 Motivation

There are three application areas that are viewed as the main sources of the need to study visibility: robotics, GIS, and computer graphics. We discuss these fields in more detail below. This overview is not intended to be complete; in the first place it tries to give a glimpse of the many appearances of visibility.

1.1.1 Visibility in robotics

Robot motion planning and robotic surveillance raised numerous different visibility challenges which, once described geometrically, posed many interesting theoretical problems. The very first visibility problem was actually motivated by security issues: where do you need to place cameras, and how many of them, in order to guard your valuables? More details on this so-called *Art Gallery Problem* are given in Section 1.3.1.

The best-known use of visibility in robotics is the use of the visibility graph in exact robot motion planning [117, 120, 146]. The visibility graph is a graph in which each vertex represents a (point) location, and if two locations can see each other, then an edge is drawn between them. Such an edge corresponds to a straight visible path from one location to the other. This path is in fact the shortest path between these two locations. In a simple version of the exact robot motion planning problem, an environment with obstacles and the source and target position for a point robot are given. The algorithm computes the visibility graph of the vertices of the obstacles, then finds a visible visibility graph vertex for both the source and the target position, finds the shortest path in the visibility graph between these two vertices, and then combines all the information to obtain a collision-free path. With the output-sensitive visibility graph construction algorithm by Ghosh and Mount [90], it is possible to find a collision-free path in $O(n \log n + K)$ time, where n is the total number of vertices of the objects, and K is the number of edges in the visibility graph.

Another important topic in robotics is the exploration of, and navigation in, (partially) unknown environments, see [52, 187] and several chapters in [33, 48, 189] for an overview of problems and techniques. Obviously, visibility plays an important role herein, since we must see something to be able to explore it. By examining the view for a robot at different times and positions, the environment becomes known little by little. Exploring such an unknown environment often yields a (2- or 3-D) model that can be used later for other purposes, and this technique is therefore also called map or model building [8, 127, 199]. A strongly related topic is that of visibility-based localization: given the view around a robot, determine where it is located in a known environment [116, 188, 198]. Often, these two problems are tackled at the same time, which is studied in the important subfield in robotics called SLAM: Simultaneous Localization and Mapping [126, 177, 187, 189].

Yet another related topic is motion planning with visibility constraints [118, 184]. Such problems are usually set in dynamic environments. This problem has many variants, but the common characteristic is that both motion and visibility constraints need to be satisfied, and often another criterion, such as path length, can be optimized. This visibility-constrained motion planning has applications in, among others, pursuit-

evation problems [85, 95, 144, 167], in which one wants to find and track a (hiding) target and first achieve and then maintain visibility. Many of these problems also have applications in games, in both serious and recreational gaming.

Visibility is often modeled theoretically by the existence of a simple line-of-sight. Therefore, there are many problems that consider other aspects, but in fact correspond to visibility issues. An example of such is that a group of (very basic) mobile robots, performing a task together, might need to be mutually visible to communicate with each other and coordinate their movement and actions. Roboticists may not use the word visibility here, but essentially it is the same thing.

Finally, we need to mention optical sensors, which play an important role in robotics, since this is often the main source of input for a robot. Physically different sensors (e.g., with limited visibility in terms of range or angle) lead to different models of observers in theoretical visibility research. There have been both theoretical [119] and applied [85] studies of such observers, but there are not that many. Because sensors are often very basic in their possibilities and have tight performance restrictions, exact algorithms are not very well applicable for them. In this context, computer vision is much more important for robotics than computational geometry.

1.1.2 Visibility in GIS

In Geographic Information Science (GIS)[1], visibility computations are referred to as viewshed analysis. The *viewshed* is the area of land that is visible from a given viewpoint, which is frequently also called viewer or observer. Computations related to, or using the viewshed, are called viewshed analysis.

Applications in GIS for which viewshed analysis is essential are numerous. A recent (and local) example is a study of the visibility of the Belle van Zuylen-tower. This huge tower might be built in Leidsche Rijn, to the west of the city of Utrecht, and should be finished by 2013 or 2014. If realized, this tower will measure 262m and thus will have a significant impact on the surroundings. The Netherlands Institute for Spatial Research (RPB) determined, with the aid of viewshed analysis, that this tower would in extreme cases be visible at a range of 62 kilometers, which means people would be able to see it from as far as Amsterdam and 's Hertogenbosch, as well as from the outskirts of Arnhem and from Den Haag [166]. This study falls into the GIS-category that is relatively negatively named *horizon pollution*. Of course, there are much more applications of viewshed analysis in GIS. Many of them are from urban planning and landscape architecture, but also from fields such as archaelogy or mobile (phone) communication. In the latter, it is generally assumed that a mobile phone has to be visible to set up a connection with a signal transmitter located somewhere in the area.

We have not yet mentioned one of the most important fields that uses visibility analysis: the military. Of course, maintaining visibility of the enemy, or in other cases,

[1] GIS is also an acronym for Geographic Information System, which is a system for storing and analyzing spatial data. These systems are an important research topic in Geographic Information Science.

staying out of view from them, is essential in conflict situations. Determining where to set up a military base will largely be dependent on the surroundings; e.g., one important issue is the ability to detect hostiles approaching your base. Besides observation issues, (tele)communication is very important in the military and, as we mentioned above, is often strongly related to visibility.

Many algorithms exist for viewshed computation [69, 70, 77, 113, 124, 178]. Some of them consider the viewshed for a single viewpoint, others compute the overlay of the viewsheds of multiple viewpoints, and yet others consider different view elements than single points. The main issue regarding visibility computations in GIS is that the amount of data in GIS is very large, and thus the computation time and the algorithm efficiency are essential. However, it's important to realize that GIS data usually is an approximation of the real world — literally. Therefore, the vast majority of GIS visibility algorithms are heuristic, approximate, or both [68, 75]. Other studies take advantage of special capabilities of (graphics) hardware to do part of the computations [45]. This is a huge contrast with research in computational geometry, where the objective of almost all visibility studies is to obtain an exact algorithmic solution.

1.1.3 Visibility in computer graphics

Computer graphics is concerned with the mathematical and computational foundations of image generation and processing. Visibility computations occur mostly in the subfield of rendering. It is most recognizable in techniques that attempt to speed up the rendering pipeline, and in particular to speed up the rasterization step. But since visibility is strongly related to illumination, the techniques that are used in the computations of lighting and shadows, also involve visibility issues. The main objective of the use of visibility computations in graphics is to obtain fast and efficient techniques that give well-rendered images, i.e., images that look realistic. We refer the reader to the survey by Brittner and Wonka [32], which gives an extensive overview of visibility problems in computer graphics.

In graphics, the real or an imaginary world is modeled by three-dimensional objects. This model needs to be *rendered* to be shown on a (computer) screen, which is of course two-dimensional. During rasterization, it is determined which 3D objects should be shown on screen, or in other words, which objects are closest to the image plane. These are exactly those objects that are visible when the projection center is the viewpoint. Such techniques are commonly called either *visible surface determination* (VSD) or *hidden surface removal* (HSR) algorithms. These techniques can either operate in image space, which means that the visible object is determined for each screen pixel, or in object space, which means that the visibility computations are performed in the three-dimensional environment model. The first techniques are often quite naive and easy to implement, and they are usually pretty fast. The latter techniques are usually more time-demanding: many algorithms take quadratic time in the number of vertices of the objects [53, 131], or depend on the number of intersections between the projected edges of the objects [94, 141, 148, 169]. Algorithms with running times that are sensitive to the size of the output exist as well [15, 19, 165, 174], usually for special kinds of

input [25, 96, 110]. Techniques in object space have the advantage that they are screen- and resolution-independent. The result of hidden surface removal in object space is in fact what is known in computational geometry as the visibility map.

An early survey of hidden surface removal techniques in image space can be found in [185]. The two best-known image space techniques are the Z-buffer algorithm and the painter's algorithm. They both essentially sort the objects by distance to the image plane and as such do not really take advantage of visibility information. There is a large group of HSR-algorithms that use Binary Space Partitions (BSP) [79, 80, 154]. They subdivide the 3D scene and store this subdivision in a tree in order to provide an unambiguous depth ordering of the 3D objects from any point in the scene. Visibility is implicitly bypassed in this technique. Another well-known hidden surface removal technique called *ray tracing* does in fact use visibility [92, 158, 176]. As the name suggests, rays are traced from the viewpoint through the scene. Ray tracing does not remove hidden surfaces as such, but it does detect visible surfaces: it determines the nearest surface along each view-ray.

To obtain realistic images, it is common to define light sources in 3D scenes and compute what effect the lighting has on the objects of the scene. In general, (parts of) objects that are visible from the light source are illuminated, and the 'hidden surfaces' lie in the shadow. The simplest light source is a point light source, which corresponds to an observer located at that given point. Computing what parts of the scene are illuminated by a given point light source is exactly the same as computing the visibility map of the scene for that given light source as the viewpoint. We refer the reader to the survey by Woo et al. [196] for an overview of the many shadow algorithms that have been developed over the years. Light transport theory describes how illumination in a scene gets from one place to another, and it is not hard to see that visibility plays an essential role in these techniques. Again, ray tracing is a common technique, where the rays now play the role of light paths as opposed to viewing directions. In this setting however, the rays are traced repeatedly, and are assumed to bounce off objects and continue their path through the scene, illuminating more than one object indirectly by reflection. The single action of detecting the first object hit by a given ray is also studied quite a lot in computational geometry, where it is often referred to as ray shooting [19]. Lighting techniques that are common in computer graphics are much more sophisticated than computing merely direct illumination and generally involve reflection and/or scattering, which describes how the light interacts with a surface at a given point, and shading, which takes the material properties into account. An early survey on such techniques is [4], but the best source for detailed information on most lighting and shadow techniques is the abundance of graphics textbooks that is available nowadays.

1.2 Visibility problems in computational geometry

The objectives in studying and using visibility computations are fundamentally different in CG than in the application fields described above. In robotics, GIS, and computer graphics, visibility is a tool: it is used to improve other computations and algorithms,

whereas in CG, it is a research topic in itself. The objectives of visibility research in CG are to unveil the computational complexity of a variety of visibility problems, and to develop, if possible, efficient algorithmic solutions to these problems.

In this section, we give an overview of what sort of visibility problems have been studied in CG in the past. More than two decades ago, a book was published that was solely dedicated to art gallery problems [149]. Since then, a number of surveys [175], handbook chapters [10, 151, 190], and Ph.D. theses with survey-like chapters [31, 57, 65] have been written about 2D and 3D visibility and illumination problems, algorithms, and applications.

In parallel, we also try to describe what characterizes a visibility problem, in terms of input and definitions. We first differentiate visibility problems based on the modeling of the input domain. These modeling choices can be based on the motivation of the problem that comes from its application(s), but they can also be purely theoretically driven. Next, we summarize the mathematical and geometric foundations of visibility. We give common definitions of visibility and mention the problems that these different settings yield. Finally, we give an overview of the three major types of visibility problems that have been studied, and mention the most important obtained results.

1.2.1 Modeling the world

In reality, a person looks around in the real world. In computational geometry, an observer looks around in an environment that is modeled by geometric objects. Environments will differ based on the setting of the problem at hand.

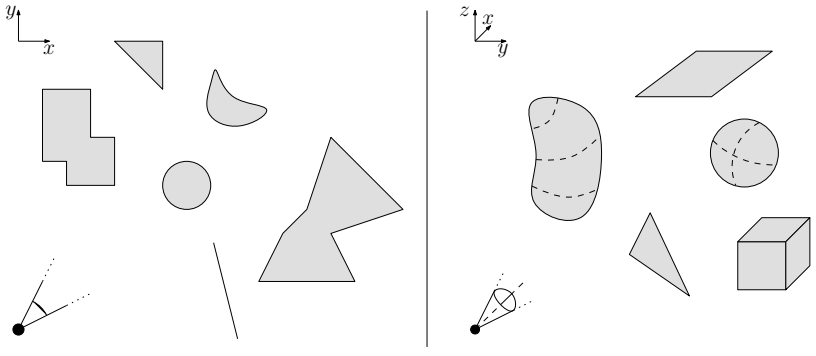


Figure 1.1: Example of a 2D (left) and a 3D (right) environment.

The most important problem parameter is the dimensionality of the space in which the computations are performed. One-dimensional visibility does not make much sense, so the easiest setting is visibility in the plane. In this case, the environment $\mathbb{E}$ will be pair consisting of a domain $\mathbb{D}$ (e.g., $\mathbb{R}^2$) and a set $\mathbb{S}$ of objects contained in this domain. The objects in $\mathbb{S}$ can range from very simple, (e.g., points, line segments, squares, disks) via objects with linear description complexity (e.g., simple polygons,

either with or without holes), to the other extreme of very complex (e.g., arrangements of high-dimensional algebraic curves). We study visibility in the plane in Chapter 2 where the environment is a simple polygon, which possibly contains holes.

The 3D equivalents of the above mentioned 2D geometric primitives can be used to model a 3D environment, which we call a *3D scene* in this thesis. Thus, in three-dimensional space, the environment can consist of points, lines, cubes, spheres, polyhedra (convex or nonconvex), or even arrangements of high-dimensional algebraic surfaces. One of the easiest modeling paradigms, which we use throughout this thesis, is a set of triangles in $\mathbb{R}^3$, which is also very common in computer graphics software and hardware. A special case of a 3D scene is the polyhedral terrain, which in some respect is a 2.5-dimensional input domain. We study polyhedral terrains in Chapters 3, 4, and 5, as well as general 3D scenes in Chapter 4.

Now that we know how to make a model environment that represents the world, and in particular what our options are to model the obstacles that obstruct seeing everything in this world, we can define visibility within this environment, which can be visibility of, and among, the objects.

1.2.2 Modeling visibility

For a human, seeing the world around him or her is very intuitive, even though the physical process that makes it possible to see is very complex. It is necessary to perform a significant abstraction to geometrically model visibility in its many appearances. In this section, we try to give an overview of the different definitions that have been devised over the years.

In the simplest setting, an observer is a single point p in the environment $\mathbb{E}$, which we also call the viewpoint. The general definition of visibility from p is the following. Given a point q in $\mathbb{D}$, we say that q is visible from p (and conversely, p is visible from q) if and only if the line segment pq does not intersect the interior of any object in $\mathbb{S}$. By stating that the *line-of-sight* pq may intersect the boundary of the objects in $\mathbb{S}$, we allow grazing contact of the line-of-sight with the objects, which is a common assumption in visibility and art gallery literature [149]. Note that such a viewpoint has the ability to look around for the whole 360°, and even look in any direction at the exact same moment, which is of course quite unrealistic in most applications.

This definition can be used in geometric environments of any dimensionality, as long as the objects in these environments have an interior and a boundary. In three-dimensional space with a set of cubes as objects, for instance, we still say that a point p and q are visible to each other if the line segment pq does not intersect the interior of any cube. The environment domain $\mathbb{D}$ can also be a (simple) polygon P in the plane, with holes in P being the objects $\mathbb{S}$. In this case, visibility is only permitted in the interior of P.

Of course, the objects limit the visibility throughout $\mathbb{E}$. Somewhere in the environment, there is a boundary between a visible and an invisible region. This boundary is defined by a so-called *critical set*. Imagine a two-dimensional environment $\mathbb{E}$ with domain $\mathbb{R}^2$ and the set of objects $\mathbb{S}$ being a a set of line segments. Somewhere in $\mathbb{E}$, the

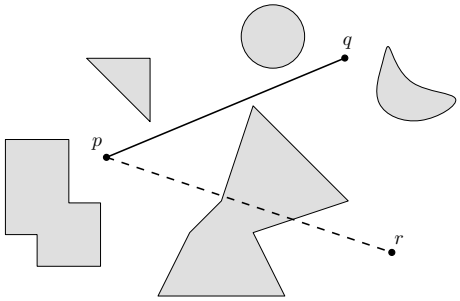


Figure 1.2: The line segment between the points p and q in the plane is free from intersections: p and q are visible to each other. The point r is not visible from p.

viewpoint p is located. Take any point not on a line segment, and take the ray from p through that point. Suppose that this ray does not intersect any line segment. Now imagine that you rotate this ray around p, until you hit one of the line segments. It is easy to see that (part of) the critical line ℓ that we obtain in this way is the transition between a visible and an invisible subset in $\mathbb{E}$; see Figure 1.3. The concept of a critical set exists in three dimensions as well. Of course now it is not a line anymore; a viewpoint and a line segment (e.g., the edge of an object) define a (part of a) critical plane.

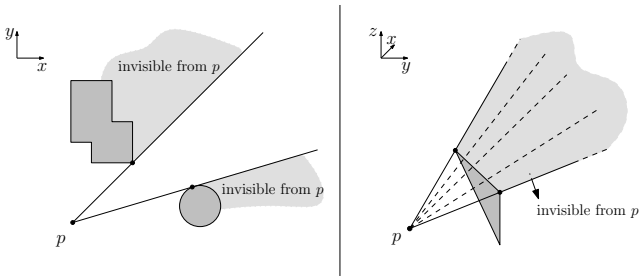


Figure 1.3: Examples of critical sets. In the left figure two critical lines are shown, and in the right figure a critical plane.

The observer (also called view element) might also consist of more than a single point. The reason to allow, for example, a line segment or a polyline as a view element is that time is implicitly captured: a chain of line segments can represent the path that a watchman walks through a building, where every point on this chain corresponds to a different location of the watchman over time.

These higher-dimensional view elements lead to two new notions of visibility: strong (complete) visibility and weak (partial) visibility. The definitions are quite straightforward. A set S strongly sees a point q in $\mathbb{E}$ if and only if every point in S sees q, and S

weakly sees q if and only if there is at least one point in S that sees q. These definitions can be used symmetrically: a viewpoint p can strongly (or weakly) see a set S.

These definitions in turn give rise to more complex critical sets; with a single viewpoint, the number of degrees of freedom is immediately restricted. If the observer is a line segment in a 3D scene consisting of triangles, for instance, the critical set is a set of critical lines, which are all tangent to three triangle edges. These lines lie on a ruled surface called a *regulus* or *swath* [57, 164]. It is easy to see that also for a spherical object and a (single) viewpoint, the critical set is a curved surface.

Critical sets are particularly important in analytical visibility computations, which we discuss in Section 1.3.3. Moreover, they are used extensively in computations concerning the visible region of a given view element, on which we elaborate in Section 1.3.2. Finally, they are often used for (intuition on) developing lower bound constructions; see for example [131], and also Chapter 4. Critical sets are also called *visibility events* [57]. A visibility event that is defined by an object vertex and an object edge is called an EV event, and one defined by three object edges is called an EEE event. These concepts are also well known in computer graphics, and are essential for the aspect graph structure [112]. More details on these critical sets, their exact definition and their locus, are given in Section 3.4.1 of Chapter 3 and in Section 4.2 of Chapter 4.

Some other definitions of visibility exist that have not been studied very extensively, although these in fact seem to model reality much better than the single viewpoint with omnivisibility that is most common in CG studies. Examples of such adapted visibility definitions are visibility within a limited angle [82, 119, 150, 190] and visibility within a limited range [147]. There are yet some other visibility definitions that seem a little bit exotic. It may not seem very natural to classify these problems as visibility problems, mainly because usually the visibility in these studies is not direct, i.e., not involving a simple straight line-of-sight. This does of course conflict with the intuitive human understanding of the concept of visibility. These definitions may still be of use for other applications, however. An example of such an exotic type of visibility is rectilinear visibility, in which we are allowed to see around corners of 90° [47, 82, 97, 105]. There are many more examples and variants that we do not wish to mention here.

1.3 Types of problems

It all started back in 1973 with the famous 'Art Gallery Problem', which was posed by Victor Klee: 'How many point guards are always sufficient to guard any simple polygon with n vertices?'. This problem classifies as a *guarding problem*, which we discuss below. There is a second class of visibility problems, which is different in nature from the guarding problems, namely those related to the computation of the set of all visible points for a given view element. The third and last class of problems that we describe is that of analytical visibility computations. This classification is not exhaustive, but we believe that with a little bit of effort, most problems that deal with visibility, will feel at home in one of the three classes.

1.3.1 Guarding problems

In this class of visibility problems, usually no viewpoints or elements are given. It is in fact the challenge to find those. Above, we already mentioned the mother of all visibility problems, which was solved by Chvátal [43] in 1975. He showed that for any simple polygon $\lfloor \frac{n}{3} \rfloor$ (point) guards are always sufficient, and sometimes necessary. There has been an abundance in studies of variations of the problem; instead of giving a long list of references here, we refer the reader to the books and surveys that were mentioned earlier in this chapter.

Some of the guarding problems merely ask for the number of guards that is needed for a given polygon, others also ask for placements of these guards. It has been known for a long time that finding the (locations of) the minimum number of guards for a given polygon is NP-hard [121], and more recently there have also been several inapproximability results [65]. For example, Eidenbenz et al. have shown that finding a $(1+\epsilon)$-approximation of the minimum guard set is NP-hard [66]. The first known exact solution to the minimum point guard problem was given by Efrat and Har-Peled [63]; their algorithm has running time $O((nc)^{3(2c+1)})$, where is c is the size of the optimal solution. Note that the running time is exponential in the size of the optimal solution, which can be $\Theta(n)$ in the worst case.

The inapproximability results did not discourage the CG community: over the years, many studies have been conducted to develop approximation algorithms. In [88], Ghosh presents an $O(\log n)$-approximation algorithm for the problem where guards can be be located only at the vertices of the polygon. Efrat and Har-Peled [63] give a randomized algorithm which with high probability achieves an approximation ratio of $O(\log c)$, where c is again the size of the optimal solution. Recently, a pseudopolynomial time $O(\log c)$-approximation algorithm for the minimum point guard problem was presented [51].

In 3D, there has been little research on guarding, and most of it considers guarding polyhedral terrains [34, 35, 63, 65, 66].

When we extend the guards to (chains of) line segments, we get a whole new subclass of problems. The view elements are now commonly called *watchman tours*. This set of problems is an extension of the art gallery setting: not cameras are placed, but a watchman is hired to guard the valuables by walking a route through the gallery. Common objectives in watchman tour problems are to minimize the length of the path [145], to minimize the number of links [5], or finding a tour under various restrictions [147].

Another subclass is that of polygon classification based on visibility properties. These problems are formally not guarding problems, but in nature, they deal with the same concepts. Popular such problems are to determine whether a given polygon is internally (or alternatively: externally) weakly visible [28, 29, 168], or whether there are two points s and t on the boundary such that they subdivide the boundary into two mutually weakly visible chains. The latter property is called L-R-visibility [30, 50].

In Chapter 2, we study a problem that falls into this class: the witness problem. We classify the problem we study in Chapter 3 in this category as well, since it determines intervisibility of sets on terrains, which will mostly be interesting if one of the sets is a guard set. The other chapters deal with the type of problem that we discuss below.

1.3.2 Computing what is visible

In the *visibility computation* problems, an environment with objects that have n vertices in total, and a view element are given, and the challenge is to compute what part of the environment is visible.

In 2D, we call the visible subset of the plane the *visibility region* of a view element e. If the environment consists entirely of polygonal objects, this region is generally called the *visibility polygon*. Computing the visibility polygon of a given point can be done in $O(n)$ time in simple polygons [107]. For more complex view elements, the visibility region also gets more complex. The (weak) visibility region of a line segment, for example, can have complexity $\Omega(n^4)$ in the worst case [183]. The strong visibility region however, still has just $O(n)$ worst-case complexity [10].

In 3D environments that consists of all polyhedral objects, the visible spatial subset for a given view element is called the *visibility polyhedron*. The set of visible parts of the (boundaries of the) objects in a general 3D scene, or sometimes the subdivision of the objects that is induced by this visible set, is called the *visibility map*. For sets of n triangles in 3D, this structure has complexity $\Theta(n^2)$ in the worst case [131]. The upper bound is obtained by projecting all object edges onto a sphere that surrounds the complete environment and counting intersections, and the lower bound is derived from the construction in Figure 1.4.

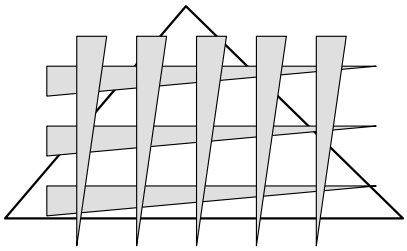


Figure 1.4: $\Omega(n^2)$ lower bound construction of the visibility map among triangles in 3D.

In some way, visibility is not really taken into account in the upper bound by McKenna [131], since the computed complexity is the complexity of the so-called *transparent* visibility map. Usually, people will be interested in the case that the objects in the environment do in fact obstruct visibility, which results in the *occluded* visibility map. This leads us back to hidden surface removal, which we discussed in Section 1.1.3. The most efficient algorithms to compute the visibility map are in fact the previously mentioned hidden surface removal techniques [15, 19, 24, 110, 152, 165, 174], and therefore we do not discuss them any further here. We study the worst-case complexity and the construction of visibility maps of several different types of view elements in Chapter 4.

We do want to highlight a recent result by de Berg and Gray [17], who give an algorithm to compute the visibility map for so-called *fat* (non-skinny) objects. Assumptions on the input are usually called *realistic input models*, which allow for faster

algorithms and improved complexity or combinatorial bounds. Many of the early results obtained with realistic input models are concerned with union complexities of fat objects (e.g. [6, 62, 130]) or motion planning in realistic environments [1, 21, 181, 182]. Two surveys of the different available models, and a hierarchy between these models, can be found in [22] and in Vleugels' thesis [192]. We present a realistic input model for polyhedral terrains in Chapter 5, which we use to show better bounds and algorithms for, among others, the visibility map.

Computing what is visible (or illuminated) in a single given direction is done by ray tracing, which is the same as repeated ray shooting. Many ray shooting algorithms are collected in the book by De Berg [15] and the survey by Pellegrini [156]. Usually, the environment is preprocessed to construct a data structure that can answer ray shooting queries efficiently [19]. There have been many results on this technique, for varying types of environments, such as sets of triangles [155], convex objects [109], or fat objects [16, 18]. Moreover, there has been a number of studies that consider restrictions on the direction of the rays being shot; the most results are available on vertical rays [16, 18, 109].

Related to the visibility map is the shadow map, where the view element is replaced by a light source. Now visibility is replaced by illumination, and invisibility by shadow. This change of names comes from the applications in computer graphics. For the shadow map, it is possible to assume that some objects are not completely opaque, and will let some light go through. This increases the complexity of the problem significantly and most results will be approximate or heuristic algorithms, as is common in computer graphics, where computations are under tight time restrictions anyway. Again, we refer the reader to the computer graphics textbooks for references. We do mention one variation here however. The common assumption is to have a point as a light source, but there are also algorithms available to compute the shadow for an area light source [55, 186]. These studies are strongly related to our results in Chapter 4.

1.3.3 Analytical data structures

We call the final class of visibility problems the *analytical visibility* problems. In these problems, an environment without a viewpoint (or view element) is given and it is often the objective to construct a complex data structure that can handle varying types of visibility queries. These data structures are very general in the sense that they capture a lot of information about the environment, and are therefore used in many different fields of research.

The first such data structure is the visibility graph of a set of objects. It was introduced by Nilson for use in motion planning problems [146]. A collection of algorithms for its construction exists [120, 195, 9]. The most efficient and output-sensitive algorithm is the one by Ghosh and Mount [90], which we already mentioned in Section 1.1.1.

Later, the visibility complex was proposed by Pocchiola and Vegter [162, 163]. This is a structure that has the same size as the visibility graph, but contains more information, which allows for shortest path and ray shooting queries. This visibility complex was later extended to the three-dimensional visibility complex [59], which is assumed

to contain 'all visibility information of a 3D scene' [57], and to the simpler and more practical visibility skeleton [58, 60]. More recently, a hybrid of the visibility complex and the Voronoi diagram was presented [194], which is intended to be used in motion planning applications.

The most important, i.e., the most used, analytical visibility structure is without a doubt the aspect graph, which was introduced by Koenderink and van Doorn [111, 112]. It captures information on all possible views of a given (3D) environment, by partitioning the space into regions throughout which the view is combinatorially the same. Its complexity depends on the type of objects in the scene. It was originally developed in computer vision for model-based object-recognition, and has received ample attention from researchers in that area [64, 160, 161]. It has been used in robotics as well, mainly in localization and pursuit-evasion applications. Computing the aspect graph is not easy or straightforward, and many have studied this complex task, especially for different types of environment [91, 157, 159]. We use the aspect graph extensively in this thesis, in particular in Chapters 3 and 4.

1.4 Contributions of this thesis

This thesis is compiled of chapters that correspond to four articles (already published or appearing soon) and one technical report. The chapters are almost direct copies of the published articles; only the lay-out has been adapted and the bibliographies have been merged into one.

Chapter 2 tries to find a discretization of a simple polygon to aid computations for guarding problems [49]. We define a *witness set* W to be a set of points in a simple polygon P such that if any (prospective) guard set G guards W, then it is guaranteed that G guards P. We first analyze the combinatorial properties of (the elements of) a witness set, and then we give an algorithm to compute a minimum size witness set for P in $O(n^2 \log n)$ time, if such a set exists, or to report the non-existence within the same time bounds. We also outline two algorithms that use a witness set for P to test whether a (prospective) guard set sees all points in P.

Chapter 3 computes whether two sets of triangles on a polyhedral terrain are mutually visible (under varying definitions) [134]. Suppose we are given a terrain T with n triangles and two two simply connected subsets R_1 and R_2 of at most m triangles of T. We present an algorithm that determines, for any constant $\epsilon > 0$, within $O(n^{1+\epsilon}m)$ time and storage whether or not R_1 and R_2 are completely intervisible. We also give an $O(m^3n^4)$ time algorithm to determine whether every point in R_1 sees at least one point in R_2. Finally, we present an $O(m^2n^2 \log n)$ time algorithm to determine whether there exists a pair of points $p \in R_1$ and $q \in R_2$, such that p and q see each other.

Chapter 4 presents upper and lower bounds on the worst-case complexity of visibility maps for different view elements among sets of n triangles in 3D, and on polyhedral terrains [135]. We show that the trivial $\Omega(n^2)$ lower bound for the worst-case combinatorial complexity of the strong visibility map of a line segment or a triangle is almost tight: we prove an $O(n^2\alpha(n))$ upper bound for both structures, where $\alpha(n)$ is the extremely slowly increasing inverse Ackermann function. Furthermore, we prove that the

weak visibility map of a line segment has worst-case complexity $\Theta(n^5)$, and the weak visibility map of a triangle has worst-case complexity $\Theta(n^7)$. For a polyhedral terrain, the worst-case complexity of the weak visibility map is $\Omega(n^4)$ and $O(n^5)$, both for a segment and a triangle. We also present efficient algorithms to compute all discussed structures.

Chapter 5 develops a realistic input model for polyhedral terrains, and uses this model to improve complexity bounds of the visibility map and two distance structures [137]. We develop a set of realistic assumptions on edge length ratios and the angles of the triangles, and a more general low-density assumption. We show that the visibility map of a point for a realistic terrain with n triangles has worst-case complexity $\Theta(n\sqrt{n})$. We also prove that the shortest path between two points p and q on a realistic terrain passes through $\Theta(\sqrt{n})$ triangles in the worst case, and that the bisector of p and q has worst-case complexity $O(n\sqrt{n})$. We use these results to show that the shortest path map for any point on a realistic terrain has complexity $\Theta(n\sqrt{n})$ in the worst case, and that the Voronoi diagram for any set of m points on a realistic terrain has worst-case complexity $\Omega(n + m\sqrt{n})$ and $O((n + m)\sqrt{n})$. Our results immediately imply more efficient algorithms for computing the various structures on realistic terrains.

Finally, Chapter 6 considers the experimental verification of the theoretical results in Chapter 5 [139]. We perform experiments with real-world data to determine how realistic the assumptions of the input model are, by examining what the values of the model parameters are in practice. We consider the results of these experiments as evidence that the assumptions of the model are probably realistic: they are all satisfied by the terrains that we generated from GIS data. Moreover, we evaluate the complexity of the visibility map in these terrains by counting vertices of the visibility maps for different viewpoints, thus collecting evidence that it may be unlikely to find a visibility map that has significantly higher complexity than the bounds in Chapter 5.

Chapter 2

Guarding art galleries by guarding witnesses

The contents of this chapter have been published as:

K.-Y. Chwa, B.-C. Jo, C. Knauer, E. Moet, R. van Oostrum, and C.-S. Shin.
Guarding art galleries by guarding witnesses.
International Journal of Computional Geometry & Applications,
R. Fleischer (Ed.), Special Issue: Selected Papers from the 15th Annual International Symposium on Algorithm and Computation (ISAAC'04) Vol. 16. No. 2/3, pages 205–226, June 2006.

Abstract

Let P be a polygon, possibly with holes. We define a *witness set* W to be a set of points in P such that if any (prospective) guard set G guards W, then it is guaranteed that G guards P. Not all polygons admit a finite witness set. If a finite minimal witness set exists, then it cannot contain any witness in the interior of P; all witnesses must lie on the boundary of P, and there can be at most one witness in the interior of every edge. We give an algorithm to compute a minimum size witness set for P in $O(n^2 \log n)$ time, if such a set exists, or to report the non-existence within the same time bounds. We also outline two algorithms that use a witness set for P to test whether a (prospective) guard set sees all points in P.

2.1 Introduction

Visibility problems have been studied extensively in the computational geometry literature, and the so-called *Art Gallery Problems* form an important subcategory within this field. The problem of how many points or *guards* are always sufficient to guard any simple polygon with n vertices was posed by Victor Klee in 1973. Chvátal [43] showed soon thereafter that $\lfloor \frac{n}{3} \rfloor$ guards are always sufficient, and sometimes necessary. Since then, there have been numerous results on variations of the problem, summarized in O'Rourke's book in 1987 [149]. Newer results appear in the surveys by Shermer [175] and Urrutia [190].

Most of the papers related to the Art Gallery Problem consider the computation of the location of the guards under various restrictions on the type of visibility (i.e., orthogonal visibility, π-lights, etc.) or on the type of guards (i.e., edge guards, vertex guards, mobile guards, etc.). Except for the case of a single guard, only few papers consider the computation of the guarded region for a *given* set of points [41, 86].

Joseph Mitchell posed the *Witness Problem* to Tae-Cheon Yang during a research visit of the latter: "Given a polygon P, does it admit a *witness set*, i.e., a set of objects in P such that any (prospective) guard set that guards the witnesses is guaranteed to guard the whole polygon?" As with the original Art Gallery Problem, there are many possible variants of this problem. The simplest kind of witnesses are points, possibly constrained to lie on the vertices or on the boundary of P. The idea, of course, is that in the case of a set of moving guards, or of a guard set that permits the addition or removal of guards, it is easier to check point-to-point (i.e., guard-to-witness) visibility, than to update the complete visibility region of all guards. Other possible types of witnesses are edges of the polygon. Yang showed this problem to a student, and they made an initial effort to classify polygons that can be witnessed by guards placed at vertices at the polygons, and polygons that can be witnessed by (partial) edges of the polygon; see [197] (in Korean).

In this chapter, we consider point witnesses that are allowed to lie anywhere in the interior or on the boundary of the polygon. We want to determine for a given polygon P whether a finite witness set exists, and if this is the case, to compute a *minimum size* witness set. The main contribution of this chapter is the combinatorial/geometrical

result that a finite minimum size witness set for a polygon (if it exists) contains only witnesses on the boundary, and there are at most $n-2$ of them, where n is the number of vertices of the polygon. Furthermore, we show that for any $n \geq 4$ there is a finitely witnessable polygon that requires at least $n-2$ witnesses. The results are not trivial: there are polygons that are not witnessable by a finite set of witnesses. There are also polygons for which there are witness sets that witness the boundary, but not the whole interior of the polygon.

For completeness, we also outline an algorithm to compute a minimum size witness set for a polygon if such a set exists, or to report the non-existence otherwise. Generating a point set that is a witness set for a polygon if the polygon admits a finite witness set takes $O(n)$ time; checking whether the polygon is finitely witnessable indeed takes $O(n^2 \log n)$ time. Also, we show how to use witness sets for testing if a set of (prospective) guards actually sees the entire polygon.

This chapter is organized as follows. In the next section we introduce the formal definition of *witness sets*, and we prove some interesting basic properties of them. In Section 2.3, we present a number of examples to give the reader a better understanding of the problem. Using the results from Section 2.2, we study properties of *finite* witness sets in Section 2.4. In Section 2.5, we discuss the cardinality of *minimum size* witness sets. In Section 2.6, we give algorithms for computing and using minimum size witness sets, and we wrap up in Section 2.7 with a brief discussion on our results and on open problems.

2.2 Preliminaries

Throughout this chapter, P denotes a polygon with n vertices. We allow P to have holes, but its boundary has no self-intersections. We say that a point p lies in P if p lies in the interior of P (denoted with $int(P)$) or on its boundary (denoted with ∂P), i.e., we consider P to be a closed subset of $\mathbb{E}^2$. We assume that the vertices $V(P)$ of P are in general position, that is, no three vertices of P are collinear[1]. The edges of P are denoted by $E(P)$. Geometrically, we consider the edge e to be the closed line segment between its incident vertices, i.e., an edge includes its endpoints. Every edge e is directed in such a way that the interior of P lies locally to the left of e.

A point p in P *sees* a point q in P if the line segment pq is contained in P. Since polygons are closed regions, the line-of-sight pq is not blocked by grazing contact with the boundary of P; this definition of visibility is commonly used in the Art Gallery literature [149]. We say that a point p in P *sees past* a reflex vertex v of P if p sees v, and the edges incident to v do not lie on different sides of the line through p and v, i.e., one of the edges may lie on this line.

Let e be an edge e of P. Let $\ell(e)$ be the directed line through e such that $\ell(e)$ has the same orientation as e. The positive halfspace induced by $\ell(e)$ is the region of points in $\mathbb{E}^2$ to the left of $\ell(e)$, and we denote it by $\ell^+(e)$. The negative halfspace $\ell^-(e)$ is defined analogously. The *closure* of a (possibly open) region of points $R \subset \mathbb{E}^2$ is the

[1] This assumption is only used in the proof of Lemma 2.5.1.

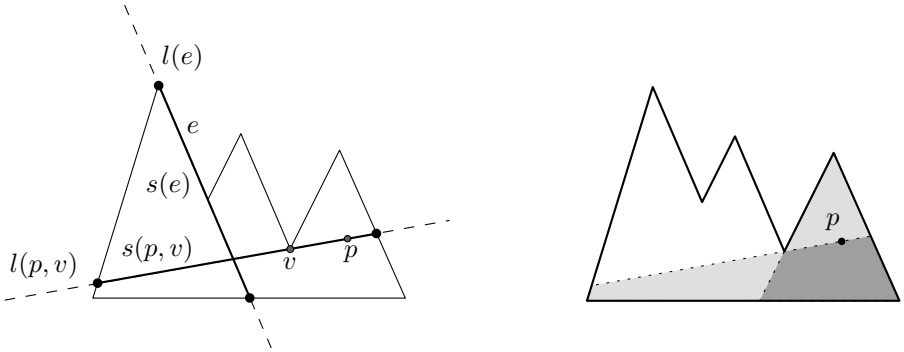


Figure 2.1: (a) Illustration of the definitions of $\ell(e)$, $s(e)$, $\ell(p, v)$ and $s(p, v)$. (b) The union of the light and dark shaded regions is $\mathrm{VP}(p)$; the dark shaded region is $\mathrm{VK}(p)$.

union of R and its boundary ∂R; we denote it with $cl(R)$.

For a point p in P and a reflex vertex v of P such that p sees past v, $\ell(p, v)$ is defined as the line through p and v. $\ell(p, v)$ is directed in such a way that the two edges incident to v both lie in $cl(\ell^-(p, v))$.

For an edge $e \in E(P)$, $\ell(e) \cap P$ consists of one or more connected parts (segments) that lie completely in P or on its boundary. The segment that contains e is denoted by $s(e)$. Similarly, for a point p in P and a reflex vertex v of P such that p sees past v, $\ell(p, v) \cap P$ consists of one or more connected parts (segments) that lie completely in P or on its boundary. The segment that contains p and v is denoted by $s(p, v)$; see Figure 2.1 for an illustration of these definitions.

The *kernel* of a polygon P is the set of points from which every point in P is visible. If the kernel is non-empty, we call P *star-shaped*. It is well-known that the kernel of a polygon P can be computed as $\bigcap_{e \in E(P)} cl(\ell^+(e))$.

Let p be a point in P. The *visibility polygon* of p is the set of points in P that are visible to p. We denote the visibility polygon of p by $\mathrm{VP}(p)$. We define the *visibility kernel* of a point p to be the kernel of its visibility polygon and we denote it by $\mathrm{VK}(p)$. The visibility polygon $\mathrm{VP}(p)$ of a point p in P is star-shaped by definition (the kernel contains at least p).

Definition 2.2.1. *A* witness set *for a polygon P is a point set W in P for which the following holds: if, for any arbitrary non-empty set of points G in P, each element of W is visible from at least one point in G, then* every *point in P is visible from at least one point in G.*

The elements of G in the above definition are commonly referred to as *guards*, and we call the elements of a witness set *witnesses*. Note that a guard set always is non-empty, but that a witness set is allowed to be empty (namely, the empty set is a witness

set for any convex polygon).

The following theorem states the necessary and sufficient conditions on witness sets.

Theorem 2.2.1. *Let P be a polygon, and let W be a point set in P. Then W is a witness set for P if and only if the union of the visibility kernels of the elements of W covers P completely.*

Proof: Let $W = \{w_1, \ldots, w_k\}$ be a set of points in P such that $\cup_{w \in W} \mathrm{VK}(w)$ covers P completely. Let G be an arbitrary set of points in P such that every element $w_i \in W$ is visible from at least one $g_j \in G$. Since g_j lies in $\mathrm{VP}(w_i)$, we have that all points in $\mathrm{VK}(w_i)$ are visible from g_j. Since there is such a g_j for every w_i, and $\cup_{w \in W} \mathrm{VK}(w)$ covers P completely, it follows that every point in P is visible from at least one $g_j \in G$, and therefore W is a witness set for P.

Conversely, let W be an arbitrary witness set for P. Let us assume for the sake of contradiction that $\cup_{w \in W} \mathrm{VK}(w)$ does *not* cover P completely.

Let us first consider the case where the union of the visibility *polygons* (as opposed to the visibility *kernels*) of all elements of W do not cover P completely. In this case, a contradiction is easily derived: place a guard g_i at every witness w_i. Now every witness is seen by at least one guard, but the guards do *not* see the whole polygon, so W is not a witness set for P.

It remains to consider the more interesting case where the union of the visibility polygons of all elements of W *does* cover P completely. Then we pick an arbitrary point p in the region of P that is *not* covered by any of the visibility kernels of the witnesses. For any $w_i \in W$, p may or may not lie in $\mathrm{VP}(w_i)$, but in either case, since p does not lie in $\mathrm{VK}(w_i)$, p cannot see *all* points in $\mathrm{VP}(w_i)$. This means that for each $w_i \in W$ we can place a guard g_i in $\mathrm{VP}(w_i)$ such that g_i does *not* see p. So every witness w_i is seen by at least one guard (namely, g_i), but the guards do *not* see every point in P (for none of the guards sees p). This means that W is not a witness set for P, and we have a contradiction again. ∎

In Section 2.3, several examples of polygons are discussed; we either show a finite witness set for the polygon, or we illustrate why no such set exists for that particular polygon.

We also apply the concept of witnesses to individual points. For two points p and q in a polygon P, we say that p is a witness for q (or alternatively, that p witnesses q), if any point that sees p also sees q. The following lemma is analogous to Theorem 2.2.1, and we omit the proof, which is much simpler than the proof of the theorem.

Lemma 2.2.1. *Let P be a polygon, and let p and q be points in P. Then p witnesses q if and only if q lies in* $\mathrm{VK}(p)$.

The following two lemmas show that witnessing is transitive.

Lemma 2.2.2. *Let P be a polygon, and let p and q be points in P. Then p witnesses q if and only if* $\mathrm{VP}(p) \subseteq \mathrm{VP}(q)$.

Proof: If a point p witnesses a point q, then $q \in \mathrm{VK}(p)$ by the preceding lemma. This implies that everything that is visible from p is also visible from q, which means that $\mathrm{VP}(p) \subseteq \mathrm{VP}(q)$.

Conversely, if $\mathrm{VP}(p) \subseteq \mathrm{VP}(q)$, then any point that sees p, and therefore lies in $\mathrm{VP}(p)$, also lies in $\mathrm{VP}(q)$, and consequently sees q. ■

Lemma 2.2.3. *Let P be a polygon, and let p, q, and r be points in P. If p witnesses q and q witnesses r, then p witnesses r.*

Proof: If p witnesses q and q witnesses r, then by the preceding lemma, $\mathrm{VP}(p) \subseteq \mathrm{VP}(q)$ and $\mathrm{VP}(q) \subseteq \mathrm{VP}(r)$. This means that $\mathrm{VP}(p) \subseteq \mathrm{VP}(r)$ and thus that p witnesses r. ■

This leads to the notion of *minimal witness sets.*

Definition 2.2.2. *Let P be a polygon, and let W be a witness set for P. W is called a* minimal witness set *for P if, for any $w \in W$, $W \setminus \{w\}$ is* not *a witness set for P.*

Lemma 2.2.4. *Let P be a polygon, and let W be a witness set for P. W is a minimal witness set for P if and only if for any $w \in W$, w does not lie in $\mathrm{VK}(w')$ for any $w' \in W, w' \neq w$.*

Proof: Because W is a witness set for P, we have that $\bigcup_{w_i \in W} \mathrm{VK}(w_i)$ is equal to P. Now suppose, for a contradiction, that there is a $w \in W$ such that, for some $w' \in W$, w lies in the visibility kernel of w'. This, by Lemmas 2.2.1 and 2.2.3, implies that all the points that are witnessed by w are also witnessed by w', i.e., $\mathrm{VK}(w) \subseteq \mathrm{VK}(w')$. But then $\bigcup_{w_i \in W \setminus \{w\}} \mathrm{VK}(w_i)$ also covers P completely, and thus W is not a minimal witness set.

Conversely, suppose that W is *not* a minimal witness set for P. This implies that there exists a $w \in W$ such that $W \setminus \{w\}$ is still a witness set for P, i.e. $\bigcup_{w_i \in W \setminus \{w\}} \mathrm{VK}(w_i)$ covers all points of P. In particular, w is a point in P, therefore there exists a $w' \in W$ such that $w \in \mathrm{VK}(w')$. ■

Observation 2.2.1. *Let P be a polygon, and let W be a witness set for P. Then (i) there exists a subset $W' \subseteq W$ such that W' is a minimal witness set for P, and (ii) for any superset $W'' \supseteq W$, W'' is a witness set for P.*

As for the second claim: note that, although adding witnesses to a witness set W may seem to place additional constraints on the placement of the guards, this is actually not the case. If the guards see all witnesses in W, then they already see the whole polygon, including any additional witnesses.

Lemma 2.2.5. *For a star-shaped polygon P and any point p in P, $\mathrm{VK}(p)$ contains the kernel of P.*

Proof: Any arbitrary point p in P sees all points q in the kernel of P, and this holds no less for any point (i.e. guard) that can see p. Therefore, p is a witness for all points q in the kernel of P, and by Lemma 2.2.1 it follows that all the points q in the kernel of P lie in $\mathrm{VK}(p)$. ■

Lemma 2.2.6. *Let P be a polygon, and let p and q be points in P. If p does not see q, then there is a reflex vertex $v \in V(P)$ located on a shortest geodesic path from p to q such that p sees past v.*

Proof: Because p does not see q, there exists a shortest geodesic path from p to q that consists of more than one line segment. Note that if P has holes, the shortest path is not necessarily unique. A shortest geodesic path from p to q has consecutive vertices $u_0 = p, u_1, \ldots, u_k, u_{k+1} = q$, where for each $1 \leq i \leq k$ the vertex u_i is a reflex vertex of P [123], and $k \geq 1$. This means that p sees past the reflex vertex u_1. ■

Lemma 2.2.7. *Let P be a polygon, and let p and q be points in P. If q lies outside* $\mathrm{VK}(p)$*, then q sees past at least one reflex vertex $v \in V(P)$.*

Proof: If P is star-shaped, then q cannot lie in the kernel of P, since by Lemma 2.2.5, q would lie in $\mathrm{VK}(p)$, which contradicts the premise of this lemma. So in this case, there must be a point q' in P that is not visible from q.

Otherwise, if P is not star-shaped, there is also a point q' that is not visible from q.

In both cases, by the preceding lemma, there is a reflex vertex $v \in V(P)$ such that q sees past v. ■

Lemma 2.2.8. *If a point p in a polygon P sees past a reflex vertex $v \in V(P)$, then p lies on the boundary of* $\mathrm{VK}(p)$*.*

Proof: The lemma follows easily from the facts that (i) p lies on $\ell(p, v)$, (ii) $\mathrm{VK}(p)$ is contained in $cl(\ell^+(p, v))$, and (iii) p lies in $\mathrm{VK}(p)$. ■

By combining Lemmas 2.2.4, 2.2.7 and 2.2.8, we get the following lemma.

Lemma 2.2.9. *Let P be a polygon. If W is a minimal witness set for P, with $|W| > 1$, and w is an element of W, then w lies on the boundary of* $\mathrm{VK}(w)$*.*

2.3 Examples

To illustrate the concept of witnesses and to provide the reader with some intuition for the problem, we give a few examples in this section. We introduce the examples informally: we do not prove any claims that we make about them. If the reader wishes to check the claims, she may use the lemmas and theorems that we give in Sections 2.2 and 2.4.

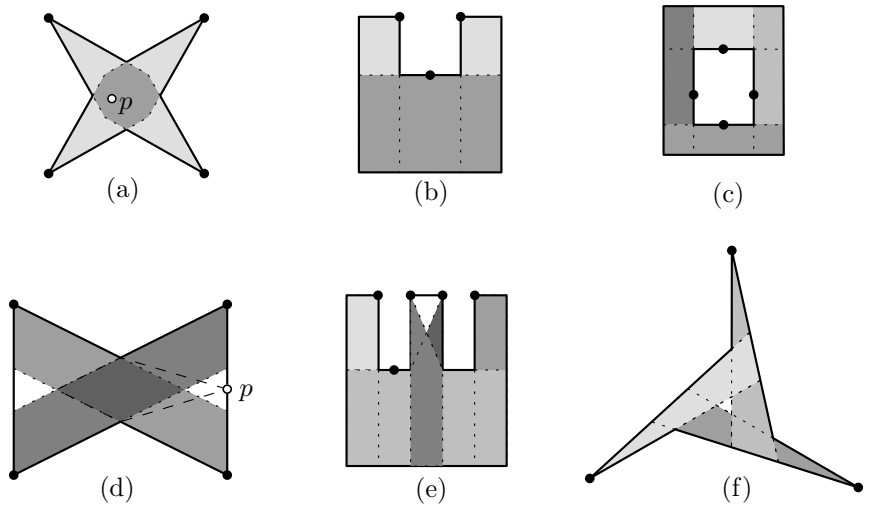


Figure 2.2: Examples of the regions in polygons witnessed by vertices and points on edges. Note that the white regions in polygons (d), (e) and (f) are not holes, but unwitnessed regions.

Our examples are given in Figure 2.2; the first five polygons are simple polygons (note that the white regions are not holes, but unwitnessed regions), and the last one is a polygon with a hole.

In Figure 2.2(a) we see a star-shaped polygon. Clearly, a point p in the kernel guards the whole polygon. One may mistakenly believe that a single point in the kernel is a witness set for the polygon as well, since p sees the witness, and p also guards the polygon.

This, however, is incorrect. According to Definition 2.2.1, *any* set of points that sees the witnesses must see the whole polygon. If p plays the role of witness, a single point in one of the arms of the polygon (i.e., outside the kernel of the polygon) sees p, but it does not see the whole polygon. Therefore, $\{p\}$ is *not* a valid witness set. The polygon of Figure 2.2(a) can be witnessed by the points shown in black.

It is not true that a polygon has to be star-shaped to be witnessable by a finite witness set; see Figure 2.2(b), which shows a polygon that is witnessable by a set of three witnesses. One of the witnesses is placed in the interior of an edge. In fact, this polygon is not witnessable by a witness set that has only witnesses placed at vertices.

Figure 2.2(c) shows a polygon with a hole, and a witness set for this polygon with four elements. All the witnesses are located in the interior of edges, and, like the polygon of Figure 2.2(b), this polygon is not witnessable by just vertex witnesses.

It is not true that all star-shaped polygons can be witnessed by a finite witness set. Figure 2.2(d) shows a star-shaped polygon, and a set of four points placed on vertices of the polygon. Their visibility kernels do not cover the whole polygon; the white regions

are not witnessed.

Adding witnesses at the remaining two vertices does not help, because these are already witnessed and witnessing is transitive. To witness the white regions, we need to place witnesses that lie on the boundary of the polygon and in the white regions. Consider such a witness p. Its visibility kernel is the convex hull of the kernel of the polygon (darkly shaded) and p itself, so the only point on the boundary that is witnessed by p is p itself. It turns out that we need to cover both unwitnessed segments on the boundary of the polygon completely with witnesses to get an (infinite) witness set for this polygon.

Figure 2.2(e) shows a polygon that is similar to the polygon in Figure 2.2(b), but which is not finite witnessable. The two convex vertices of the vertical dent in the center both see past a reflex vertex, and thus the edge in between them is not witnessed.

Figure 2.2(f) shows an example of a polygon that cannot be witnessed with a set of points on the boundary; even if we would cover the whole boundary of this polygon with witnesses, the white region in the middle would remain unwitnessed. A witness set for this polygon would have to include all points in that region (and each point in the region only witnesses itself).

2.4 Finite witness sets

In this section we bound the number of witnesses of a finite minimal witness set W for a polygon P from above. We first show that the elements of W can only lie on the boundary of P (Lemma 2.4.1), and next, that any edge of P has at most one element of W in its interior (Lemma 2.4.5).

We denote the regions of points in P witnessed by any of the elements of a set S of points in P by $\mathcal{R}(S)$. If S is finite, then $\mathcal{R}(S)$ consists of one or more *closed* polygonal regions, since it is the union of a finite set of kernels of visibility polygons, which are closed. Note that in degenerate cases, a region in $\mathcal{R}(S)$ may be a single point or a line segment. The regions of points that are *not* witnessed by any of the elements of S are denoted by $\mathcal{Q}(S) = P \setminus \mathcal{R}(S)$, and these are called the *unwitnessed regions*. $\mathcal{Q}(S)$ consists of one or more connected (but not necessarily simply connected) polygonal regions that are either open (if the region lies in $int(P)$) or neither open nor closed (if the region contains part of ∂P; see Figure 2.3). The important fact is that no part of $\partial\mathcal{Q}(S)$ that lies in $int(P)$ belongs to $\mathcal{Q}(S)$ itself.

Lemma 2.4.1. *Let P be a polygon. If W is a finite minimal witness set for P, then no element of W lies in $int(P)$.*

Proof: If W is the empty set, then the lemma holds trivially.

W cannot have only one element. Otherwise P would be convex, and since the empty set is also a valid witness set for convex polygons, this would contradict the minimality of W.

It remains to show that the lemma holds in case W has more than one element. In this case, consider any witness $w \in W$. Note that $\mathcal{Q}(W \setminus \{w\})$ cannot be a point or a one-dimensional region; otherwise, $\mathcal{R}(W \setminus \{w\})$ would not be closed, and this is only

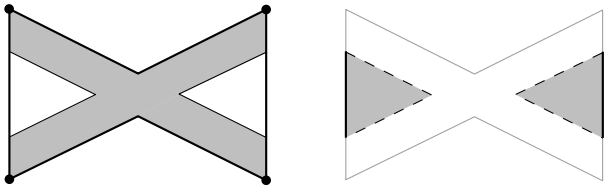


Figure 2.3: Since the union of a set of visibility kernels (left) is closed, the resulting unwitnessed regions (right) may be neither open nor closed.

possible if W is infinite. Also note that, because $\mathcal{R}(W)$ is equal to P, the unwitnessed regions $\mathcal{Q}(W \setminus \{w\})$ are strictly contained in $\mathrm{VK}(w)$. From Lemma 2.2.4 we deduce that w lies in $\mathcal{Q}(W \setminus \{w\})$. Since no part of $\partial\mathcal{Q}(W \setminus \{w\})$ that lies in $int(P)$ belongs to $\mathcal{Q}(W \setminus \{w\})$ itself, w can only lie in $int(\mathcal{Q}(W \setminus \{w\}))$ or on $\mathcal{Q}(W \setminus \{w\}) \cap \partial P$. By Lemma 2.2.9, w lies on the boundary of $\mathrm{VK}(w)$. Since w cannot lie simultaneously on the boundary of $\mathrm{VK}(w)$ (which is closed) and in the open set $int(\mathcal{Q}(W \setminus \{w\}))$, the conclusion is that w can only lie on $\mathcal{Q}(W \setminus \{w\}) \cap \partial P$. ■

Lemma 2.4.2. *Let P be a polygon, and let W be a finite minimal witness set for P. Then no element of W lies on a reflex vertex of P.*

Proof: Suppose, for the sake of contradiction, that there is an element w of W that lies on a reflex vertex v of P. Let e and e' be the edges of P incident to v. Observe that, because $\mathrm{VK}(w)$ is contained in $cl(\ell^+(e)) \cap cl(\ell^+(e'))$, $\mathrm{VK}(w)$ does not contain any points on e and e', except v itself. This means that all points on e and e' (except v) must be witnessed by the elements of $W \setminus \{w\}$. However, since W is finite, the union of the visibility kernels of the other witnesses is a closed region, and therefore v is also covered by these visibility kernels. This means that w is witnessed by another witness $w' \in W$, and this contradicts the minimality of W. ■

We established that witnesses of a finite minimal witness set W lie on the boundary of P, but not on reflex vertices. Using the following two lemmas, we show in Lemma 2.4.5 that every edge of P has at most one element of W in its interior.

Lemma 2.4.3. *Let P be a polygon, let W be a finite minimal witness set for P, and let w be an element of W. If w sees past any reflex vertex v of P, then w must lie on an edge e of P such that v lies on $\ell(e)$.*

Proof: Since W is a finite minimal witness set for P, any $w \in W$ lies on some edge e of P by Lemma 2.4.1. Note that w lies on two such edges if it lies on a vertex. Now suppose, for the sake of contradiction, that w *does* see past one or more reflex vertices v of P that do not lie on $\ell(e)$. Then there is at least one reflex vertex v such that w sees past v and $\mathrm{VK}(w)$ is contained in $cl(\ell^+(w, v))$.

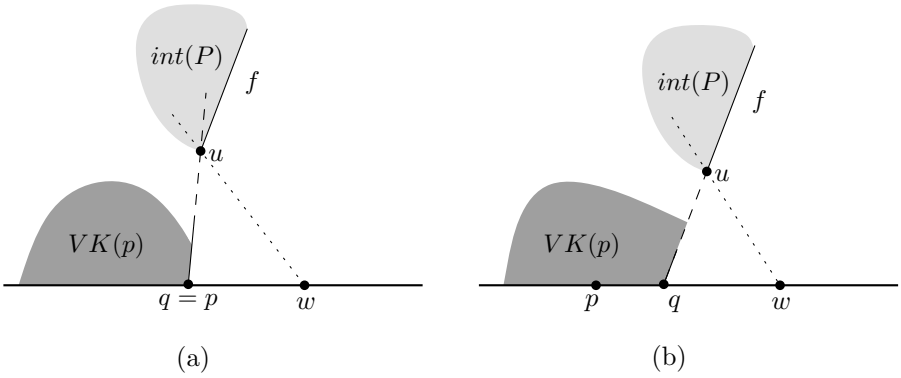


Figure 2.4: Illustrations to the proof of Lemma 2.4.4. The witness w sees past u or (not shown) past another reflex vertex in the triangle formed by w, u, and q.

We now regard only the boundary of P. The intersection of any visibility kernel (which is convex) with ∂P consists of one or more parts that are homeomorphic to a point or a to a closed line segment. Similarly, since W is finite, $\mathcal{R}(W \setminus \{w\}) \cap \partial P$ consists of one or more parts that are homeomorphic to a point or a closed line segment, and $\mathcal{Q}(W \setminus \{w\}) \cap \partial P$ consists of one or more parts that are homeomorphic to an *open* line segment.

The remainder of the proof is essentially a one-dimensional version of the proof of Lemma 2.4.1. Since (i) w lies on the intersection of e and $\ell(w, v)$, (ii) $\mathrm{VK}(w)$ is contained in $cl(\ell^+(w, v))$, and (iii) w sees past v, w must be the endpoint of one of the parts of $\mathrm{VK}(w) \cap \partial P$. Since W is a minimal witness set, by Lemma 2.2.4, w must also lie in $\mathcal{Q}(W \setminus \{w\})$. But w cannot lie simultaneously in an open subset and on the boundary thereof, so we conclude that w does not see past any reflex vertex that does not lie on $\ell(e)$. ∎

Under our assumption that P is in general position, the previous lemma implies that any element w of a finite minimal witness set can only see past a reflex vertex v if v is incident to the edge on which w lies. However, Lemma 2.4.3 also holds if P is not in general position.

Lemma 2.4.4. *Let P be a polygon, and let W be a finite minimal witness set for P. If a witness $w \in W$ is located in the interior of an edge e, then every point p in $int(e)$ and every convex vertex incident to e (if any), witnesses w.*

Proof: Let e be an edge of P that contains a witness $w \in W$ in its interior. The lemma is trivial for $p = w$, since every point witnesses itself.

For the sake of contradiction, let us assume that an arbitrary point $p \neq w$ in $int(e)$ or on a convex vertex incident to e does not witness w, i.e., w does not lie in the visibility kernel of p. Without loss of generality, we assume p lies to the left of w. The closed line

segment $\mathrm{VK}(p) \cap e$ has an endpoint q somewhere in between p and w. We now have two cases.

1. q is the intersection of e and $\ell(p, u)$ for a reflex vertex u such that p sees past u; see Figure 2.4(a). Let f be the edge incident to u that p does not see, and let f' be the other edge incident to u.
2. q is the intersection of e and $s(f)$ for some edge f of P, visible from p; see Figure 2.4(b). Let u be the vertex of f closest to e, and let f' be the other edge incident to u.

Note that in both cases, w does not see f, and f' cannot lie in $cl(\ell^+(f))$, because u is a reflex vertex. We again have two cases to consider.

1. If the boundary of P does not intersect the interior of the triangle formed by w, q, and u, then w sees past the reflex vertex u;
2. Otherwise, if the boundary of P does intersect the interior of the triangle formed by w, q, and u, then w does not see u and, by Lemma 2.2.6, there is a reflex vertex u' on a shortest geodesic path from w to u such that w sees past this vertex u'.

Both cases contradict Lemma 2.4.3 and thus we conclude that p witnesses w. ■

By combining Lemmas 2.2.4, 2.4.2 and 2.4.4, we get the following lemma.

Lemma 2.4.5. *Let P be a polygon, and let W be a finite minimal witness set for P. If an edge e of P contains a witness $w \in W$ in its interior, then no other element of W can lie on e.*

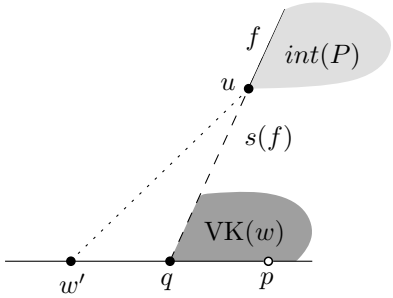


Figure 2.5: Illustration to the proof of Lemma 2.4.6. If an edge e is (partially) witnessed by a witness w not on e, then there cannot be a witness w' in the interior of e.

The next lemma is used to prove Lemma 2.4.7, stating that any witness in a finite minimal witness set that lies in the interior of an edge, witnesses that entire edge.

Lemma 2.4.7 is applied in Section 2.5.3, where we investigate the possible locations of the witnesses on the boundary of P.

Lemma 2.4.6. *Let P be a polygon, and let W be a minimal finite witness set for P. If a witness $w \in W$ that does* not *lie on an edge e of P witnesses a point in the interior of e, then there cannot be a witness $w' \in W$ that lies in the interior of e.*

Proof: Suppose that there is a witness w that does not lie on an edge e and that witnesses a point p in the interior of e. Since p lies in the interior of e, w must lie in $cl(\ell^+(e))$. Now suppose, for the sake of contradiction, that there is a witness w' in $int(e)$, and without loss of generality, assume that w' lies to the left of p. See Figure 2.5. We have that p lies in $\mathrm{VK}(w)$, but w' does not lie in $\mathrm{VK}(w)$ (otherwise, by Lemma 2.2.4, the minimality of W would be contradicted). So the boundary of $\mathrm{VK}(w)$ must intersect e in between w' and p; let the intersection point be q. This point q is actually the intersection of e and $s(f)$, where f is an edge visible from w. Let the vertex incident to f that is closest to q be u. Let f' be the other edge incident to u. Note that the boundary of P cannot intersect the interior of the segment qu, and thus f' lies in $cl(\ell^-(f))$; otherwise, $s(f)$ would not extend all the way to q. We now consider two cases.

1. If the boundary of P does not intersect the interior of the triangle formed by w', q, and u, then w' sees past the reflex vertex u;

2. Otherwise, if the boundary of P does intersect the interior of the triangle formed by w', q, and u, then w' does not see u and, by Lemma 2.2.6, there is a reflex vertex u' on a shortest geodesic path from w' to u such that w' sees past this vertex u'. Both sub-cases contradict Lemma 2.4.3.

We conclude that there cannot be a witness w' in $int(e)$. ∎

Lemma 2.4.7. *Let P be a polygon, and let W be a finite minimal witness set for P. If $w \in W$ lies in the interior of an edge e of P, then w witnesses the whole edge e.*

Proof: If w does not witness e completely, there must be at least one other witness w' that witnesses a part of e. By Lemma 2.4.5, w' cannot lie on e. We know that w' witnesses at least one point on the interior of e, since the part of e that does *not* lie in $\mathrm{VK}(w)$ cannot consist of only one or both vertices of e (because visibility kernels are closed regions). But then by Lemma 2.4.6, there cannot be a witness that lies in $int(e)$ – and this contradicts the fact that w lies in $int(e)$. ∎

Combining Lemmas 2.4.1, 2.4.5, and 2.4.7, we derive the following theorem.

Theorem 2.4.1. *Let P be a polygon with n edges. If a finite minimal witness set W for P exists, then all witnesses $w \in W$ lie on the boundary of P, and any edge that contains a witness $w \in W$ in its interior contains no other witnesses. Furthermore, a witness $w \in W$ in the interior of an edge e of P must witness e completely, as no other witness $w' \in W$ can witness any point on e.*

2.5 Minimum size witness sets

In this section, we classify the edges of a finite witnessable polygon P into three categories, depending on whether their incident vertices are convex or not: we distinguish reflex-reflex edges, convex-reflex (or reflex-convex) edges, and convex-convex edges. In Section 2.5.1, we show that each reflex-reflex and convex-reflex edge of P contains exactly one witness, and in the following section we show that no witness lies on a convex-convex edge, except possibly at its vertices that are also incident to a convex-reflex edge. In Section 2.5.3, we show *where* to place witnesses on reflex-reflex and convex-reflex edges. For each reflex-reflex edge we can place a witness anywhere in its interior, and for each convex-reflex edge we can place a witness on its convex edge. This placement strategy establishes a minimum size witness set for P—if a finite witness set for P exists. Finally, Section 2.5.4 discusses the cardinality of a witness set constructed in this way. Testing whether the candidate witness set is indeed a witness set, i.e., whether P is finite witnessable, is the subject of Section 2.6.

2.5.1 Reflex-reflex and convex-reflex edges

Lemma 2.5.1. *Let P be a polygon, let W be a finite minimal witness set for P, and let e be an edge of P that is incident to at least one reflex vertex. Then there is exactly one witness $w \in W$ located on e.*

Proof: Let v be a reflex vertex incident to e. There must be a witness w of W that witnesses a point p in the interior of e, as well as v itself, since $\mathrm{VK}(w)$ is a closed region, and we have only finitely many witnesses in W.

We first show that w must lie on $s(e)$; recall from Section 2.2 that $s(e)$ is the single piece of $\ell(e)$ that lies in P and that contains e. Suppose, for the sake of contradiction, w does *not* lie on $s(e)$. Obviously, both v and p are visible from w, so w must lie in $\ell^+(e)$, and by Lemma 2.4.1, w lies on the boundary of P. We have two cases to consider.

- w sees past v. According to Lemma 2.4.3, this is only allowed if v lies on $\ell(f)$, where f is the edge on which w lies. Because $w \in \ell^+(e)$, f cannot be incident to v, otherwise v would not be reflex. But this violates the assumption made in Section 2.2 that no three vertices of P are collinear.

- w does not see past v. Since w sees v, we have that w sees both edges e and e' incident to v and $\mathrm{VK}(w)$ is completely contained in $cl(\ell^+(e)) \cap cl(\ell^+(e'))$. But since v is reflex, this means that the only point on e that is contained in $\mathrm{VK}(w)$ is v, contradicting the fact that w also witnesses p.

Since both cases lead to a contradiction, we conclude that w lies on $s(e)$.

Since P is in general position, and w lies on the boundary of P by Lemma 2.4.1, w lies either on e or on one of the endpoints of $s(e)$. Let q be an endpoint of $s(e)$. q is either a convex vertex of e, or it does not lie on e. In the latter case, q sees a reflex vertex v' of e as well as (parts of) both edges e and e' incident to v' (v' may or may not be v). Were w located on q in that case, then we argue again that the only point

on e that is contained in $\mathrm{VK}(w)$ is v', contradicting the fact that w also witnesses p. We conclude that w lies on e.

By Lemma 2.4.2, w does not lie on a reflex vertex of e, and therefore we consider the following two cases.

- w lies in the interior of e. By Lemma 2.4.5, no other witness besides w can lie on e;
- w lies on the convex vertex of e (this case is only applicable if e is a convex-reflex edge). By Lemma 2.4.5, no other witness can lie in $int(e)$.

In both cases, w is the only witness on e and thus the proof is completed. ■

2.5.2 Convex-convex edges

No witness of a minimal witness set is located on a convex-convex edge, except possibly at a vertex that is also incident to a convex-reflex edge.

Lemma 2.5.2. *Let P be a polygon, and let W be a finite minimal witness set for P. Then no element of W is located in the interior of a convex-convex edge or on a vertex incident to two convex-convex edges.*

Proof: If W is the empty set, then the lemma holds trivially.

W cannot have only one element. Otherwise P would be convex, and since the empty set is also a valid witness set for convex polygons, this would contradict the minimality of W.

It remains to show that the lemma holds in case W has more than one element. In this case, suppose for the sake of contradiction, that there is a witness $w \in W$ that lies either in the interior of a convex-convex edge e, or on a convex vertex that is shared by two convex-convex edges e and e'. There is at least one other witness $w' \in W$, and, by Lemma 2.2.4, w lies outside $\mathrm{VK}(w')$. By Lemma 2.2.7, w sees past at least one reflex vertex v of P, and by Lemma 2.4.3 v lies on $\ell(e)$ (or on $\ell(e')$). However, all vertices incident to e (and e') are convex, so this is impossible. ■

2.5.3 Placement of witnesses

By Lemma 2.5.1, every finite minimal witness set W for P contains exactly one element w located on a convex-reflex or reflex-reflex edge. The following lemma will be useful in constructing a minimum size witness set for a polygon, and the correctness follows immediately from Lemma 2.4.4.

Lemma 2.5.3. *Let P be a polygon, and let W be a finite minimal witness set for P. Furthermore, let e be a convex-reflex or reflex-reflex edge of P, and let w be the witness located on e. If w lies in $int(e)$, then $\{w'\} \cup W \setminus \{w\}$ is also a witness set for P, where $w' \notin W$ is a replacement witness placed anywhere in $int(e)$ or on the convex vertex of e (if any).*

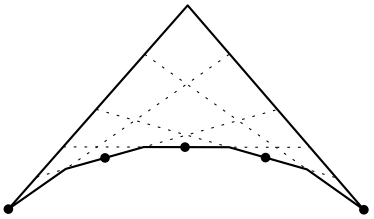


Figure 2.6: For any n there is a polygon with n vertices that is witnessable with no less than $n-2$ witnesses. Witnesses in the example are indicated with black dots.

Combining the lemmas in Sections 2.4 and 2.5 we derive our main theorem.

Theorem 2.5.1. *Let P be a polygon. If P is witnessable by a finite witness set, then a* minimum size *witness set for P can be constructed by placing a witness anywhere in in the interior of every reflex-reflex edge and by placing a witness on the convex vertex of every convex-reflex edge.*

2.5.4 Size of a minimum size witness set

A natural question is: "How many witnesses are needed to witness P when P is witnessable by a finite witness set W?". First, we observe that for every $n \geq 4$ there is a polygon that is witnessable with no less than $n-2$ witnesses, as is illustrated in Figure 2.6.

For a more general analysis, we place the witnesses on ∂P according to Theorem 2.5.1 and count them using a charging scheme. We give every witness one unit, which it pays to the reflex vertices 'close to it'.

Let us look at witnesses on convex vertices first. Such a convex vertex v has at least one incident convex-reflex edge, but possibly two. In the first case, we call v unshared, and in the latter case, we call it shared. A witness on an unshared convex vertex pays its unit to the only neighboring reflex vertex, and a witness on a shared convex vertex pays half a unit to each of its neighboring reflex vertices.

Furthermore, we have witnesses in the interior of reflex-reflex edges: they pay half a unit to each of the reflex vertices incident to that edge.

To count the number of witnesses, we sum up the total amount of units paid to all reflex vertices. Every reflex vertex has exactly two neighboring witnesses, and receives either 1 or 0.50 unit from each of them. Thus, the total income for every reflex vertex is 1, 1.50 or 2 units. In other words, a reflex vertex receives two units minus the units it might lose by sharing a witness with other reflex vertices.

For every shared convex vertex v, there are two reflex vertices that each get half a unit from the witness on v. So, there is a total loss of one unit per shared convex vertex. The same holds for every reflex-reflex edge. Furthermore, the total loss in units for a

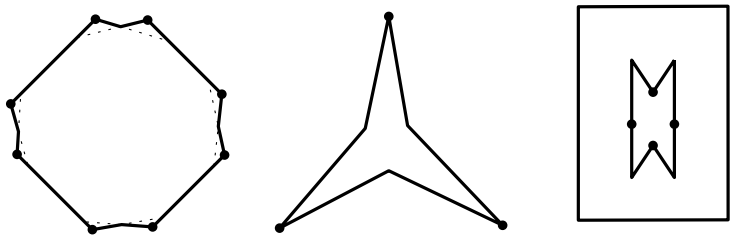


Figure 2.7: Three polygons with minimum size witness sets for them; the witnesses are indicated with black dots.

reflex vertex is at most 0.50 per incident edge, since an edge cannot be a reflex-reflex edge and at the same time lead to a shared convex vertex.

Now, by the discussion above, we have the following theorem.

Theorem 2.5.2. *Let P be a polygon, and let W be a* minimum size *witness set for P. Furthermore, let $r(P)$ be the number of reflex vertices of P, let $s(P)$ be the number of shared convex vertices of P, i.e., convex vertices that are incident to two convex-reflex edges, and let $rr(P)$ be the number of reflex-reflex edges of P. Then $|W| = 2r(P) - s(P) - rr(P)$,*

A few examples of minimum size witness sets are displayed in Figure 2.7.

2.6 Algorithms

In Section 2.6.2, we outline an algorithm that computes a minimum size finite witness set W for a polygon P, if such a set exists, or reports the non-existence of such a set otherwise. We also outline an algorithm that uses a witness set W for P to test whether a set of points G in P guards the whole polygon.

But first, we show that the arrangement induced by the visibility kernels of the witnesses has quadratic complexity.

2.6.1 Visibility kernel arrangement

In Section 2.2, the visibility kernel of a point p was described as $\bigcap_{e \in E(\mathrm{VP}(p))} cl(\ell^+(e))$. However, there is an alternative way of describing $\mathrm{VK}(p)$ that turns out to be useful.

The edges of the visibility polygon $\mathrm{VP}(p)$ can be classified into two groups.

1. An edge e of $\mathrm{VP}(p)$ coincides with the part of an edge e' of P that is visible from p. In this case, $\ell^+(e) = \ell^+(e')$. We denote the set of edges of P that are (partially) visible from p with $\mathrm{VES}(p)$.

2. An edge e of $\mathrm{VP}(p)$ is induced by the directed line $\ell(p, v)$ through p and a reflex vertex v of P such that p sees past v. The line $\ell(p, v)$ coincides with $\ell(e)$, and the orientation of the former line is defined to be the same as the orientation of the latter line. We denote the set of reflex vertices v_i of P such that p sees past v_i with $\mathrm{SPS}(p)$.

The visibility kernel $\mathrm{VK}(p)$ of p can now alternatively be described as

$$\bigcap_{e \in \mathrm{VES}(p)} cl(\ell^+(e)) \cap \bigcap_{v \in \mathrm{SPS}(p)} cl(\ell^+(p, v))$$

The reader may wonder why we introduce this seemingly complicated alternative representation of $\mathrm{VK}(p)$. The reason is that for any p, at most two vertices in $\mathrm{SPS}(p)$ suffice to define $\bigcap_{v \in \mathrm{SPS}(p)} cl(\ell^+(p, v))$. This, in turn, helps us to reduce the complexity of the data structures involved in computing the union of a set of visibility kernels.

Lemma 2.6.1. *Let P be a polygon and let p be a point in P. Then the intersection $\bigcap_{v \in \mathrm{SPS}(p)} cl(\ell^+(p, v))$ consists of only the point p itself,* or *there exists a subset $\mathrm{SPS}'(p) \subseteq \mathrm{SPS}(p)$ of cardinality at most two such that $\bigcap_{v \in \mathrm{SPS}'(p)} cl(\ell^+(p, v)) = \bigcap_{v \in \mathrm{SPS}(p)} cl(\ell^+(p, v))$.*

Proof: We argue that we can construct the intersection of halfspaces incrementally, and we prove the lemma by induction. Let the vertices in $\mathrm{SPS}(p)$ be labeled $v_1, v_2, \ldots, v_k$, and let $\mathrm{SPS}^{\mathrm{i}}(p) \subseteq \mathrm{SPS}(p)$ be $\{v_1, v_2, \ldots, v_i\}$, for $1 \leq i \leq k$.

Note that $\mathrm{SPS}^{\mathrm{k}}(p)$ is equivalent to $\mathrm{SPS}(p)$, and observe that all lines in $\{\ell(p, v) | v \in \mathrm{SPS}(p)\}$ pass through p. Let R_i be a shorthand for $\bigcap_{v \in \mathrm{SPS}^{\mathrm{i}}(p)} cl(\ell^+(p, v))$.

The lemma holds trivially for $\mathrm{SPS}^1(p)$ and $\mathrm{SPS}^2(p)$. Suppose that the lemma holds for $\mathrm{SPS}^{\mathrm{i}}(p)$, for some $2 \leq i < k$. Then we can distinguish two cases. Either R_i consists of only the point p, or it is equivalent to $cl(\ell^+(p, v')) \cap cl(\ell^+(p, v''))$, for $v', v'' \in \mathrm{SPS}^{\mathrm{i}}(p)$.

1. if R_i consists of only the point p, then $R_{i+1} = R_i \cap cl(\ell^+(p, v_{i+1}))$ also consists of only the point p, since p is contained in $cl(\ell^+(p, v_{i+1}))$ as well.

2. if R_i is equivalent to $cl(\ell^+(p, v')) \cap cl(\ell^+(p, v''))$, for $v', v'' \in \mathrm{SPS}^{\mathrm{i}}(p)$, then we distinguish three sub-cases (see Figure 2.8).

   (a) R_i lies completely in $cl(\ell^+(p, v_{i+1}))$. In that case, $R_{i+1} = R_i$, and therefore R_{i+1} is also equivalent to $cl(\ell^+(p, v')) \cap cl(\ell^+(p, v''))$.

   (b) R_i lies completely in $cl(\ell^-(p, v_{i+1}))$. In that case, R_{i+1} consists of only the point p.

   (c) R_i lies partially in $cl(\ell^+(p, v_{i+1}))$. In that case, R_{i+1} is either $cl(\ell^+(p, v')) \cap cl(\ell^+(p, v_{i+1}))$ or $cl(\ell^+(p, v_{i+1})) \cap cl(\ell^+(p, v''))$

In either case and sub-case, if the lemma holds for $\mathrm{SPS}^{\mathrm{i}}(p)$, for some $2 \leq i < k$, it also holds for $\mathrm{SPS}^{\mathrm{i}+1}(p)$, and since it holds for $\mathrm{SPS}^1(p)$ and $\mathrm{SPS}^2(p)$, we conclude that the lemma holds for $\mathrm{SPS}^{\mathrm{k}}(p) = \mathrm{SPS}(p)$. ∎

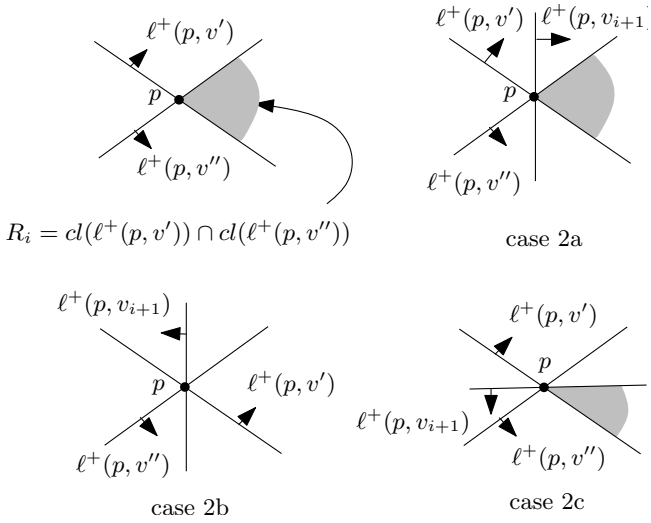


Figure 2.8: The intersection of closed halfspaces defined by lines through p is either p, or it is defined by two of the lines.

Corollary 2.6.1. *Let P be a polygon, and let W be a point set, located in P, with cardinality m. For any point p, $\mathrm{VES}(p)$ and $\mathrm{SPS}'(p)$ are defined as above. The union $\bigcup_{w \in W} \mathrm{VES}(w)$ has at most n elements, namely all edges of P. The union $\bigcup_{w \in W} \mathrm{SPS}'(w)$ has cardinality at most $2m$.*

Above, we noted that there are two types of edges for a visibility kernel: (1) the part of an edge of P that is visible from a witness w and (2) the directed line through w and a reflex vertex v of P such that w sees past v. For a polygon P and a set W of points in P, let $\mathcal{A}(W)$ be the arrangement induced by, for every witness $w \in W$, the set of line segments of type (1) and the two of type (2) that contribute to $\mathrm{VK}(w)$. For any cell c of $\mathcal{A}(W)$ and any point $w \in W$, c lies either completely inside or completely outside $\mathrm{VK}(w)$. This means that W is a witness set for P if and only if every cell of $\mathcal{A}(W)$ is contained in $\mathrm{VK}(w)$ for at least one $w \in W$. We denote the cardinality of W by m. By Corollary 2.6.1, there are in total at most $n + 2m$ line segments that define $\mathcal{A}(W)$, and therefore the complexity of $\mathcal{A}(W)$ is $O((n + m)^2)$. In Theorem 2.5.2, we showed that a minimum size witness set W has a linear worst-case complexity, thus $\mathcal{A}(W)$ has complexity $O(n^2)$ in the worst case.

2.6.2 Algorithms

The algorithm to decide whether a finite witness set exists for a given polygon P with n vertices, and to construct a minimum size finite witness set W if it exists, works as follows.

Algorithm DecideFiniteWitnessability :
INPUT: A polygon P
OUTPUT: A minimum size witness set W for P, if it exists, `False` otherwise

1. Create a candidate witness set $W = \{w_1, ..w_k\}$ by placing witnesses w_i at the positions as defined in Theorem 2.5.1.
2. Test whether W is a witness set for P:
   (a) Compute the visibility polygon $\mathrm{VP}(w_i)$ for each $w_i \in W$.
   (b) Compute the visibility kernel $\mathrm{VK}(w_i)$ from $\mathrm{VP}(w_i)$ for each $w_i \in W$.
   (c) Compute the arrangement induced by all the computed visibility kernels and by P itself, using a sweepline algorithm [13]. During the sweep, maintain for every cell in the arrangement whether it is covered by at least one visibility kernel, or only by P:
      - In the latter case, abort and return `False`.
      - Otherwise, if the algorithm is not aborted, return W.

Step 1 runs in $O(n)$ time. Step 2(a) can be performed in $O(n)$ time per witness [107], and takes $O(n^2)$ time overall. The resulting polygons have linear size each. Next, in Step 2(b), from each visibility polygon $\mathrm{VP}(w)$, we compute its kernel $\mathrm{VK}(w)$. Again, this takes linear time per polygon [122] and $O(n^2)$ time overall. The resulting kernels have linear size each. As shown in the previous section, the arrangement $\mathcal{A}(W)$ that we compute in Step 2(c) has quadratic complexity, and this step thus takes $O(n^2 \log n)$ time.

This leads to the following theorem.

Theorem 2.6.1. *Let P be a polygon with n vertices. If a finite witness set for P exists, such a finite set W of minimum size can be computed in $O(n^2 \log n)$ time. Otherwise, if no finite witness set for P exists, this can be reported within the same time bound.*

Next, we discuss approaches for testing whether a set G of g (prospective) guards in a polygon P together see the whole polygon.

A straightforward way, without using witnesses, would be to compute the visibility polygons of the g guards in $O(gn)$ total time [107]. Their union has complexity $O(gn + n^2)$, and can be computed in $O((g^2 + gn \log g) \log(g + n))$ time [41, 86]. Next, we test whether this union covers P; this can be done easily within the time bounds of the previous step, e.g., by comparing the areas of the polygons.

Can we do better if a witness set W of size m for P is given? A logical approach is to test for each witness whether it can be seen by a guard by performing (at most) g ray shooting queries, or $O(gm)$ queries in total. P can be preprocessed for ray shooting in $O(n)$, after which a query takes $O(\log n)$ time [104]. So the total preprocessing time, including the computation of W, becomes $O(n^2 \log n)$ time, and the query time is $O(gm \log n)$. Note that in the worst case $m = \theta(n)$. This query time is faster then the straightforward approach described above, but not necessarily very much (how much

precisely depends on the parameters g and m). If m is small and g is large, then the gain is big.

Alternatively, we can compute the arrangement induced by the visibility polygons of the witnesses (note: *not* the visibility *kernels*), and preprocess it for point location. This arrangement has $O((mn)^2)$ complexity (which is $O(n^4)$ in the worst case), and it can be computed and preprocessed for point location in $O((mn)^2 \log mn)$ time [13, 142], after which point location takes $O(\log(n+m)) = O(\log n)$ time per guard. If we store in each cell which visibility polygons cover the cell, then querying with g guards gets us g sets of size $O(m)$ in $O(g \log n)$ time. Then we must check whether these sets cover the set of witnesses; this can be done trivially in $O(gm)$ time. The total query time of this approach is slightly better than that of the ray shooting approach, but preprocessing time and memory requirements become much larger.

We conclude that the approaches that use witnesses are worthwhile if the polygon can be witnessed by a small number of witnesses, or if the number of guards is large.

2.7 Concluding remarks

We showed that if a polygon P admits a finite witness set, then no minimal witness set W for P has any witnesses in the interior of P, and a minimum size witness set for P can be constructed in linear time. If it is not known in advance whether P is witnessable with a finite witness set, then checking whether the constructed (candidate) witness set witnesses the polygon can be done in $O(n^2 \log n)$ time. An interesting open problem is to find a sub-quadratic algorithm for testing the candidate witness set for a polygon, or to show that this problem is 3SUM-hard.

In Section 2.2, we made the assumption that P is in general position. For some degenerate polygons, in which three vertices are collinear, and for which a finite witness set exists, a minimum witness set can be constructed by putting a restriction on our placement strategy of Theorem 2.5.1. For every sequence of collinear edges, we allow a witness on only one of these edges; see Figure 2.9(a). Other polygons that are not in general position give rise to more problems in constructing a minimum witness set, because the placement of witnesses cannot be determined locally; see Figure 2.9(b).

An interesting direction for further research is to extend the class of finitely witnessable polygons by considering other types of witnesses. We have been looking into segment witnesses, using various visibility criteria, e.g., weak versus strong, and one-sided visibility versus two-sided visibility. However, our results have been negative so far. For all of our attempts to pose conditions on the witness sets, we eventually found a polygon that admitted a segment set meeting the conditions, but that was not witnessed by the segments.

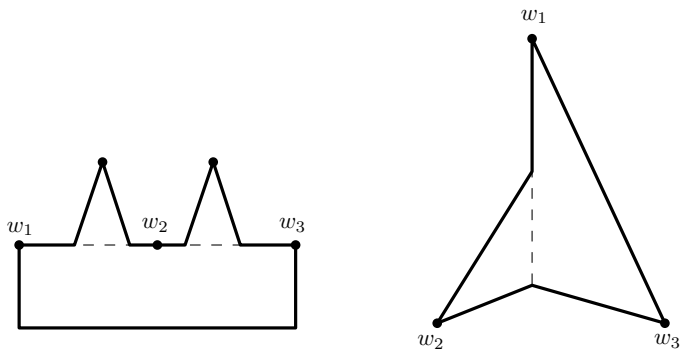


Figure 2.9: (a) Some degenerate cases can be solved easily (namely by allowing either one of the displayed witnesses for the sequence of collinear edges). (b) Other cases give more problems. The set $\{w_1, w_2, w_3\}$ is not a minimum size witness set, since $w_3 \in \mathrm{VK}(w_1)$. However, a slight perturbation of any one of the collinear vertices either makes the polygon unwitnessable, or changes $\mathrm{VK}(w_1)$ such that it does not contain w_3 any longer.

Chapter 3

Region intervisibility in terrains

The contents of this chapter have been published as:

E. Moet, M. van Kreveld, and R. van Oostrum.
Region intervisibility in terrains.
International Journal of Computional Geometry & Applications,
Vol. 17, No. 4, pages 331–347, August 2007.

Abstract

A polyhedral terrain is the graph of a continuous piecewise linear function defined over the triangles of a triangulation in the xy-plane. Two points on or above a terrain are visible to each other if the line-of-sight does not intersect the space below the terrain. In this chapter, we look at three related visibility problems in terrains. Suppose we are given a terrain T with n triangles and two regions R_1 and R_2 on T, i.e., two simply connected subsets of at most m triangles. First, we present an algorithm that determines, for any constant $\epsilon > 0$, within $O(n^{1+\epsilon}m)$ time and storage whether or not R_1 and R_2 are completely intervisible. We also give an $O(m^3n^4)$ time algorithm to determine whether every point in R_1 sees at least one point in R_2. Finally, we present an $O(m^2n^2 \log n)$ time algorithm to determine whether there exists a pair of points $p \in R_1$ and $q \in R_2$, such that p and q see each other.

3.1 Introduction

Visibility computations in 3D, especially those with respect to terrain models, have their main application in geographic information systems (GIS). Examples are computations regarding horizon pollution and signal transmission, e.g., for mobile phone networks. Other application areas in which visibility problems in 3D arise are computer graphics and game design. This research area has received increasing attention from computational geometers over the last two decades.

The most common terrain model is the *polyhedral terrain*, which is the graph of a continuous piecewise linear function defined over the n triangles of a triangulation in the xy-plane. Three surveys of a variety of visibility problems and their solutions are [44, 74, 190]. Interest in algorithmic exploration of visibility in terrains is growing. For instance, there is a number of recent articles that study how the shortest pair of watchtowers that guards a terrain can be found efficiently [2, 11, 27].

Given a viewpoint on or above a terrain, the visibility map for that point is defined as the subdivision of the terrain into visible and invisible connected components. The most efficient and output sensitive algorithms to compute the visibility map of a terrain from a given viewpoint use hidden surface removal techniques. Reif and Sen [165] developed an $O((n+k)\log n \log\log n)$ time algorithm, where k is the size of the resulting visibility map, which can be quadratic. A few years later, Overmars, Katz and Sharir [110] reported an $O((n\alpha(n) + k)\log n)$ time algorithm, where $\alpha(n)$ is the extremely slowly growing inverse of the Ackermann–function.

In GIS, visibility problems on terrains are referred to as *viewshed analysis*. Efficient computation of the viewshed, i.e., the visibility map from a given viewpoint, has been studied extensively over the years [70, 69, 77, 178, 113]. In addition, some studies addressed the relation of the height of (points on) the terrain to visibility [77, 124]. The computation of visibility in GIS is usually performed with heuristical algorithms and provides approximate solutions [68, 75]. Recently, a method of terrain simplification that aims to preserve inter-point visibility was proposed [12]. In the algorithms community, the goal is to establish the computational complexity of GIS-inspired visibility

problems and the worst-case running times of exact solutions are analyzed [26, 73]. This chapter falls into the last category.

Previous algorithmic intervisibility studies only address either point-to-point or point-to-region visibility. In contrast, we study region-to-region visibility. A region is a simply connected subset of $O(m)$ triangles of the terrain. We introduce three ways to define region-to-region visibility: complete, semi-complete, and partial intervisibility. In the case of complete intervisibility, every point in one region sees every point in the other region and vice versa. Partial intervisibility signifies that there is a point in the one region that sees at least one point in the other region. Semi-complete intervisibility is a hybrid: every point in one region sees at least one point of the other region, and vice versa.

In this chapter, we present three algorithms for the above mentioned problems. First, we determine whether two regions on a terrain are completely intervisible within time and storage bounds of $O(n^{1+\epsilon}m)$. We show that this problem is 3SUM-hard, so it is likely that our solution is close to optimal. The second algorithm we present tests for semi-complete intervisibility in $O(m^3n^4)$ time. Finally, we give an $O(m^2n^2 \log n)$ time algorithm to determine partial intervisibility.

This chapter is structured as follows. In Section 3.2 we give definitions which we use in the rest of the chapter. In Sections 3.3, 3.4, and 3.5, we consider complete, semi-complete and partial region intervisibility, respectively. In the last section, we summarize our results and give our views on future research. A preliminary abstract of this chapter appeared earlier, containing only the results of Section 3.3.1 to 3.3.3 [115].

3.2 Preliminaries

In this section, we give several definitions regarding terrains and visibility that we use throughout this chapter.

We define a *terrain* $T \subset \mathbb{R}^3$ to be a triangulated polyhedral surface with n vertices $V(T) = \{v_1, v_2, ..., v_n\}$. Each vertex v_i is specified by three real numbers (x_i, y_i, z_i), which are its cartesian coordinates. Furthermore, T can be intersected by any vertical line at most once. Therefore, T can also be viewed as the image of a piecewise linear function on $\mathbb{R}^2$. A *region* is a simply connected subset of $O(m)$ triangles of T.

We denote the plane that contains a given triangle t by $\mathcal{P}_t$. The half-space induced by $\mathcal{P}_t$ that contains the points above $\mathcal{P}_t$ is denoted by $\mathcal{P}_t^+$ and the other half-space by $\mathcal{P}_t^-$. The set of points above the entire terrain T is denoted by T^+, and the set of points below T is denoted by T^-. We assume that a triangle t of a terrain T is the union of its interior $int(t)$ and its boundary ∂t, i.e., t is a closed set. We denote the boundary of a region R by ∂R and the remainder of the points in R by $int(R)$. The closure of a (possibly open) set S is the union of S and ∂S, and is denoted by $cl(S)$. Throughout this chapter, we use $O^*(f(n,m))$ as a shorthand for the bound $O(f(n,m) \cdot n^\epsilon)$, where $\epsilon > 0$ is an arbitrarily small constant. The $O^*(..)$-notation supresses polylogarithmic factors of n, and factors n^ϵ.

Given a terrain T in 3D and two points p and q, with $p, q \in T \cup T^+$, we say that p *sees* q if the line segment pq does not intersect T^-, i.e. grazing contact of the line-of-

sight with the terrain is permitted. Visibility is only defined on and above the terrain; so, if a point q in the interior of a triangle t sees a point p, then p must lie in $cl(\mathcal{P}_t^+)$.

Definition 3.2.1. *Let R_1 and R_2 be two regions on a terrain T. R_1 and R_2 are said to be* completely intervisible *if every point in R_1 sees every point in R_2. R_1 and R_2 are* partially intervisible *if there exists a point in R_1 that sees at least one point in R_2. Furthermore, R_1 is said to* weakly see *R_2 if every point in R_1 sees at least some point in R_2. If R_1 weakly sees R_2 and R_2 weakly sees R_1, then R_1 and R_2 are* semi-completely intervisible.

Observe that the above definitions are symmetric. Also note that if R_1 and R_2 have $O(m)$ straight line segments as edges, but do not consist of whole triangles of T, then we can refine T such that both regions do consist of whole triangles, while we add no more than $O(m)$ triangles.

Throughout this chapter, we assume that two regions R_1 and R_2 on a terrain T are given, and we study complete, semi-complete, and partial intervisibility between R_1 and R_2 in Section 3.3, 3.4, and 3.5, respectively.

3.3 Complete intervisibility

In this section, we look at complete visibility between two regions R_1 and R_2; in other words, we answer the following question.

Problem CompleteIntervisibility :
Does every point in R_1 see every point in R_2, and thus does every point in R_2 see every point in R_1 as well?

First, we obtain some preliminary results. In Section 3.3.2, we describe the algorithm and the data structures we use to test for complete intervisibility. Next, in Section 3.3.3, we derive an upper bound for the problem by analyzing the running time and storage of our algorithm, and we improve this bound by a time-memory trade-off. Finally, in Section 3.3.4, we prove that CompleteIntervisibility is 3sum-hard, which indicates that our algorithm is probably close to optimal.

3.3.1 Geometric properties

To get started, we study the geometry of the problem. First, we investigate the orientations of R_1 and R_2 in space, i.e., for the time being, we ignore that other parts of T than the regions themselves can block visibility between R_1 and R_2; we remove this restriction later on. If R_1 and R_2 see each other completely, then all points in R_1 must lie in the intersection of the positive half-spaces induced by the triangles of R_2, and vice versa. Formally, the following condition has to be satisfied:

$$\forall p_1 \in R_1 : p_1 \in \bigcap_{t_2 \in R_2} cl(\mathcal{P}_{t_2}^+) \quad \wedge \quad \forall p_2 \in R_2 : p_2 \in \bigcap_{t_1 \in R_1} cl(\mathcal{P}_{t_1}^+) \tag{3.1}$$

Observe that if a point p in R_1 does not lie in the intersection of the half-spaces induced by the triangles of R_2, then at least one of the triangles of R_2 does not see p. Thus, condition (3.1) is necessary for complete visibility between two regions on a terrain. Note that it is not sufficient. We say that regions R_1 and R_2 are *facing each other* if condition (3.1) is satisfied. It is easy to verify that condition (3.1) need only be satisfied for the vertices of R_1 and R_2, in order for it to be satisfied for all points in the two regions, because $\bigcap_{t_2 \in R_2} cl(\mathcal{P}^+_{t_2})$ and $\bigcap_{t_1 \in R_1} cl(\mathcal{P}^+_{t_1})$ both are convex. Only when two regions R_1 and R_2 on a terrain T are facing each other, we have to consider the rest of the terrain as well to determine complete intervisibility. The visibility between points from R_1 and R_2 now depends on the rest of the triangles of T, i.e., those should not block the view.

Lemma 3.3.1. *Two regions R_1 and R_2 on a terrain T are completely intervisible if and only if*

1. *R_1 and R_2 are facing each other, and*
2. *∂R_1 and ∂R_2 are completely intervisible.*

Proof: The necessity of the first condition follows from the discussion above. The necessity of the second condition is easily understood. Because a region is a closed set, the boundary is a subset of it, and hence the boundaries of R_1 and R_2 have to see each other if the entire regions see each other.

To prove the sufficiency of the two conditions, we assume that 1. and 2. are satisfied, but for the sake of contradiction, we assume that there exist two points $p \in int(R_1)$ and $q \in int(R_2)$ that do not see each other, and show that this implies that there are points $p' \in \partial R_1$ and $q' \in \partial R_2$ that are not mutually visible.

Consider the vertical plane μ containing p and q. Let $T_\mu = T \cap \mu$ be the cross section of the terrain induced by μ; this is a lower-dimensional terrain itself, see Figure 3.1.

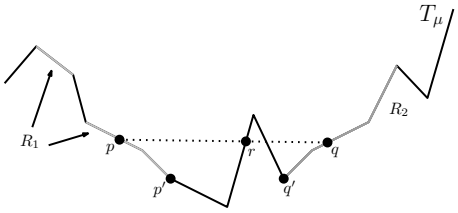


Figure 3.1: Illustration of the proof of Lemma 3.3.1.

The set $R_1 \cap \mu$ consists of one or more connected components on T_μ, and p lies in the interior of one of them; the same holds for R_2 and q. We only need to consider the visibility in T_μ to show that there exist a point $p' \in T_\mu \cap \partial R_1$ and a point $q' \in T_\mu \cap \partial R_2$ that do not see each other. Because the two triangles on which p and q lie are facing each other, the line segment pq does not intersect T^-_μ in an ϵ-neighborhood of p, nor in an ϵ-neighborhood of q. Let r be the point on T_μ closest to p that is in the closure of

$T_\mu^- \cap pq$. If p and r are in the same connected component of $R_1 \cap \mu$, then the triangle containing r is not facing the triangle containing q, which contradicts the assumption. Otherwise, there exists a point $p' \in \partial R_1 \cap T_\mu$ that lies between p and r on T_μ, such that p' cannot see q. We repeat this argument with q and p', which gives us a point $q' \in \partial R_2 \cap T_\mu$ that cannot see p'. Since $p' \in \partial R_1$ and $q' \in \partial R_2$, we have a contradiction, which proves the lemma. ■

Note that checking complete intervisibility only between the vertices of ∂R_1 and the vertices of ∂R_2 is not sufficient, because a skinny spike in the terrain can block two boundary edges from being completely intervisible, while their endpoints can indeed see each other.

Definition 3.3.1. *Let R_1 and R_2 be two regions on a terrain T. The* visibility triangle set VTS *is defined to be the set containing all triangles that are induced by either (i) a vertex of ∂R_1 and an edge of ∂R_2, or (ii) an edge of ∂R_1 and a vertex of ∂R_2.*

Because grazing contact with the terrain is permitted, intervisibility between ∂R_1 and ∂R_2 (and thus between R_1 and R_2 is blocked if and only if there is a triangle $t \in$ VTS for which the intersection $t \cap T^-$ is nonempty. Now the name *visibility triangles* becomes clear: if all triangles in VTS are free from intersections with T^-, then there is complete intervisibility between R_1 and R_2. The following lemma holds for an arbitrary triangle with its vertices in $V(T)$, so it holds for the triangles in VTS in particular.

Lemma 3.3.2. *Let T be a terrain, and let t be an arbitrary triangle with its vertices in $V(T)$. The intersection $t \cap T^-$ is nonempty if and only if at least one of the following two situations occurs:*

1. *a vertex of T lies strictly above t, or*
2. *an edge of T lies strictly above an edge of t.*

Proof: First, we observe that if $t \cap T^-$ is nonempty, then there cannot be a single terrain triangle that lies above every point in t, since the vertices of t are vertices of T, and every vertical line intersects T at most once.

Necessity: Suppose the intersection $t \cap T^-$ is nonempty, and let p be an arbitrary point in $t \cap T^-$. There is a unique point $q \in T$ that lies vertically above p; let t' be the triangle of T that contains q. Because a triangle is a closed set, we have that a vertex of t' (and of T) lies strictly above t, that an edge of t' (and of T) lies strictly above an edge of t, or both.

Sufficiency: If no vertex or edge of T is strictly above t, there are two situations: t is completely contained in T, or t intersects both T and T^+. In both cases, t does not intersect T^-. ■

3.3.2 Algorithm

In this section, we describe the algorithm to determine complete visibility between two regions in a terrain, and the data structures it uses. Using the results from Section 3.3.1, this algorithm can be described as follows:

Algorithm COMPUTE_COMPLETE_INTERVISIBILITY :
INPUT: A terrain T and two regions R_1 and R_2 on T.
OUTPUT: `true` if R_1 and R_2 are completely intervisible, `false` otherwise.

1. Compute whether R_1 and R_2 satisfy condition (3.1), i.e., whether they are facing each other; return `false` if they are not.
2. Construct the set VTS as defined in Definition 3.3.1.
3. Determine whether ∂R_1 and ∂R_2 are intervisible in two steps:
   (a) Test whether there is a vertex of T that lies above some triangle of VTS.
   (b) Test whether there is an edge of T that lies above some edge of a triangle in VTS.
4. If both step 3(a) and 3(b) yield `false`, return `true`. Otherwise, return `false`.

In the first step, we determine whether the two regions are facing each other. For each region R_i, $i = 1, 2$, we compute $\bigcap_{t \in R_i} \mathcal{P}_t^+$, which is the intersection of $O(m)$ half-spaces and can be computed in $O(m \log m)$ time [23]. Next, we check for all $O(m)$ vertices of the other region whether they are contained in $\mathcal{C}_i = cl(\bigcap_{t \in R_i} \mathcal{P}_t^+)$. Since $\mathcal{C}_i$ is unbounded in the z-direction, point location for the 3D volume $\mathcal{C}_i$ corresponds to planar point location with respect to its xy-projection. Thus, we can preprocess $\mathcal{C}_i$ for point location in $O(m \log m)$ time and then query with $O(m)$ points in $O(\log m)$ time per query [23]. In conclusion, we can check if two regions on a terrain are facing each other in $O(m \log m)$ time.

Constructing VTS can trivially be done in $O(m^2)$ time. For the third step of the algorithm (as a whole), we have to check for intersections between a set of $O(n)$ terrain triangles and a set of $O(m^2)$ triangles induced by the edges and vertices of ∂R_1 and ∂R_2. In a brute-force algorithm, this leads to $O(nm^2)$ time. However, we can do better if we perform steps 3(a) and 3(b) separately.

Step 3(a): No terrain vertex above a visibility triangle

To reduce the time complexity, it is helpful to take a different look at the geometry. For a given vertex v of the terrain, we want to check whether there is any triangle in VTS that lies strictly below v. Obviously, we want to limit the number of triangles to test against v.

If v lies vertically above a triangle t in VTS, then the projection of v onto the xy-plane lies in the projection of t. The set VTS contains $O(m^2)$ non-disjoint triangles; we want to retrieve exactly those triangles $S(v) \subseteq$ VTS that contain the projection of v in their projections. Thus, the remaining question is: does v lie below all triangles of $S(v)$? This question can also be simplified by looking at the geometry. If a point p in 3D does not lie below all triangles in a set $S(v)$, it lies outside of the polyhedron $\bigcap_{t \in S(v)} \mathcal{P}_t^-$.

To find the triangles from VTS that contain a given point in their projections on the xy-plane, we construct a partition tree of $O(m^2)$ size [23]. Given a query point p, it returns $O^*(m)$ canonical subsets of triangles from VTS that contain p in their

projection. We give every node μ in the partition tree an associated data structure of linear complexity in the cardinality of the canonical subset $c(\mu)$. This structure is used to determine whether the query point lies in the intersection of the half-spaces induced by all triangles in $c(\mu)$. This query can be answered in $O(\log m)$ time after $O(m \log m)$ preprocessing time, by using the Dobkin–Kirkpatrick hierarchy [54, 140].

The construction time for a partition tree of size $O(m^2)$ is $O(m^{2+\epsilon})$ for any constant $\epsilon > 0$ [23]. Fortunately, the space required for the partition tree's associated structure does not increase the total storage space much, in particular, nothing at all in O^*-notation. The total data structure requires $O(m^2 \log m)$ space. The construction time is $O^*(m^2)$ and the total query time is $O(n)$ times $O^*(m)$, which is $O^*(mn)$.

Step 3(b): No terrain edge above a visibility edge

We want to limit the number of terrain edges against which we check the visibility edges. We project the terrain edges onto the xy-plane and for every edge from VTS, we want to find the terrain edges whose projections intersect the projection of the query edge. Note that no edge projects to a single point. When edges of the terrain and the query edge intersect in the projection, we can treat both these terrain edges and the query edge as full lines in 3D. The objective now is to find out whether the line supporting the query edge lies above all lines supporting the selected terrain edges.

The data structure that we use stores the projections of the terrain edges onto the xy-plane in a cutting tree of size $O(n^2)$ [23]. An ordinary cutting tree stores unbounded lines and given a query point, it returns those lines that lie above the query point. Because we want to query with a line segment s_q and would like to retrieve those line segments that intersect s_q, we create a four-level cutting tree of the same size. With the first two levels, we retrieve those line segments whose supporting lines lie above one of the endpoints of s_q and below the other. To narrow this set of line segments down to the ones that actually intersect with s_q, we add another two levels in the cutting tree; these are cutting trees on the dual arrangement of the endpoints of line segments that appear in canonical subsets of the first two levels.

We perform at most m^2 queries on this tree with edges from VTS, each taking $O(\log n)$ query time. The query returns all intersecting terrain edges in $O(\log n)$ canonical subsets. We give each node μ in the cutting tree an associated structure of size $O(n^{2+\epsilon})$ for the canonical subset $c(\mu)$ of edges as the supporting lines in space [38]. Consequently, we can decide whether a query line lies above all lines in the canonical subset in $O(\log n)$ query time.

The storage required for the cutting tree and its associated structure is $O(n^{2+\epsilon})$, and the total data structure can be constructed in $O^*(n^2)$ time. The total query time is $O(m^2 \log n)$, or $O^*(m^2)$, and thus the total time complexity of step 3(b) is $O^*(n^2 + m^2) = O^*(n^2)$.

3.3.3 Running time analysis

So far, we have achieved an upper bound of $O^*(n^2)$ for problem CompleteIntervisibility, which improves the $O(nm^2)$ time of a brute-force algorithm. In Table 3.1, we

summarize the time and storage complexities for the partition tree with its associated structures (PT) and the cutting tree with its associated structures (CT).

	Preprocessing	Total queries	Storage
PT	$O^*(m^2)$	$O^*(mn)$	$O(m^2)$
CT	$O^*(n^2)$	$O^*(m^2)$	$O^*(n^2)$

Table 3.1: Summary of time and storage complexities

Now we perform a trade-off between the storage and query time to improve the upper bound to $O^*(mn)$ time.

In step 3(b), we determined whether any terrain edge lies above any visibility edge by constructing an $O^*(n^2)$ size cutting tree with an associated structure for lines in space on the terrain edges. To make the time complexity more dependent on m than on n, we create $k = n/m$ groups of terrain edges of size n/k each and build k of the augmented cutting trees described above, one for each group. This implies that every structure has size $O^*(\frac{n^2}{k^2}) = O^*(m^2)$ and the total complexity of all structures is $O^*(\frac{n^2}{k}) = O^*(nm)$. If we query these k structures with a single visibility edge, this takes $O(k \log \frac{n}{k})$ time, so for m^2 queries, this takes $O(m^2 k \log \frac{n}{k}) = O^*(nm)$ time. This is the same time complexity as we need to perform step 3(a), so there is no need to improve that step as well (even though we could perform a time-memory trade-off there to achieve a running time of $O^*(n^{\frac{2}{3}} m^{\frac{4}{3}} + n)$).

By putting the results of Sections 3.3.3 to 3.3.4 together, we get the following result.

Theorem 3.3.1. *Let T be a terrain, and let R_1 and R_2 be two regions on T, with $O(m)$ triangles each. Complete intervisibility of R_1 and R_2 can be determined in $O^*(nm)$ time and space.*

3.3.4 3SUM-hardness

In this section, we prove that CompleteIntervisibility is at least as difficult as the well-known problem 3sum.

Problem 3sum :
Given a set S of m integers, are there three elements of S that add up to 0?

The best known algorithm for this problem takes $\Theta(m^2)$ time. If a problem is at least as difficult to solve as 3sum, we say that it is 3sum-*hard*, which implies that there is little hope of finding a subquadratic time algorithm for that problem. In [81], a large collection of problems is described, with proofs that all of these problems are 3sum-hard. One of these problems is GeomBase.

Problem GeomBase [81] :
Given a set of m points S with integer coordinates on three horizontal lines $y = 0$, $y = 1$, and $y = 2$ in the plane, determine whether there exists a non-horizontal line containing three of the points.

We give a reduction from GeomBase to our problem and thus prove that CompleteIntervisibility is 3sum-hard. To achieve this, we first assume that grazing contact of the line-of-sight between two points is not permitted; later, we show how we can relax this restriction to obtain the reduction in our original setting. Assume we are given an instance S of the problem GeomBase. We transform this into an instance of CompleteIntervisibility, such that the answer we obtain by solving CompleteIntervisibility is the same as the answer we would have obtained had we solved GeomBase directly. Let A be those points of S that lie on the line $y = 0$, let B be those on the line $y = 1$, and C those on $y = 2$. The aim is to create a terrain T in which there is *no complete intervisibility* if and only if *there exists a line* through a point in A, one in B and one in C.

We construct T by placing two copies of a steep slope opposite each other. We place the two regions R_1 and R_2 on these slopes; the regions are both similar, namely a rectangular shape with a number of small protrusions at the bottom. This construction is displayed in Figure 3.2. The vertices of the protrusions of both R_1 and R_2 are located at the same height value, say $z = 1$. The vertices of R_1 are located at the x- and y-coordinates of the $O(m)$ points in A and the vertices of R_2 at the x- and y-coordinates of the $O(m)$ points in C. In between R_1 and R_2, we place a number of skinny peaks with their summits at the x- and y-coordinates of the $O(m)$ points in B and height $z = 1$. Finally, we connect all subconstructions such that they form a terrain.

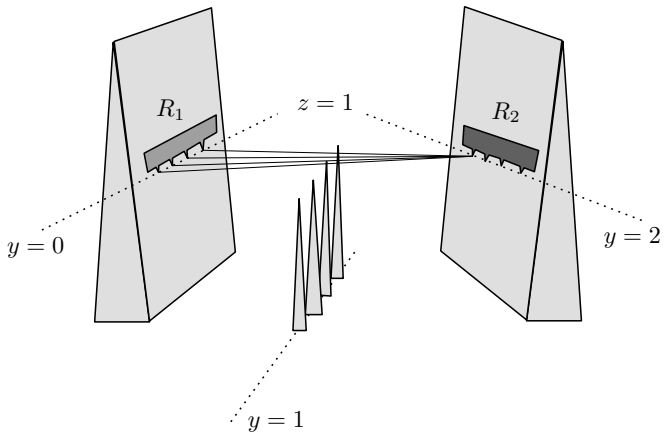


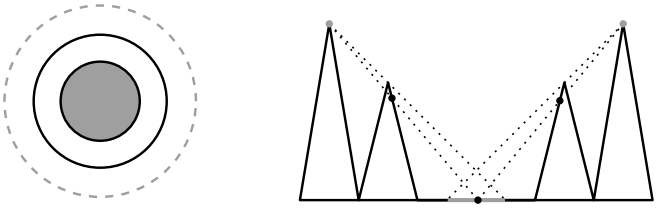
Figure 3.2: Reduction of GeomBase to CompleteIntervisibility.

Now it is clear that if and only if the answer to GeomBase is “no", the answer

to CompleteIntervisibility is "yes", i.e., the regions R_1 and R_2 are completely intervisible. The reduction takes $O(n \log n)$ time, since we connect everything to obtain a single terrain, which involves triangulating the different parts of the construction. Because we did not allow grazing contact of the line-of-sight in this reduction, we have to adjust our construction such that it also is valid if grazing contact is permitted. To achieve this, we symbolically say that the summits of the peaks that correspond to the points in B are infinitesimally higher whenever we compare them in height to another feature of the terrain. Therefore, we have a valid reduction from GeomBase and our problem is 3sum-hard. Since $\Omega(n)$ is also an (obvious) lower bound for CompleteIntervisibility, we conclude that it is unlikely that an optimal algorithm to determine complete region intervisibility has a running time of $o(m^2 + n)$.

3.4 Semi-complete intervisibility

We next present an algorithm to determine whether two regions R_1 and R_2 are semi-completely intervisible. Semi-complete intervisibility is much less restricted than complete intervisibility. For example, the triangles in the set VTS (from Definition 3.3.1) can be intersected by the terrain many times while the two region boundaries are still semi-completely intervisible. We expect an algorithm to determine semi-complete intervisibility to be more complex and time-consuming than an algorithm to determine complete intervisibility. Furthermore, contrary to the approach of the previous section, computing only intervisibility between the boundaries of the two regions is not sufficient; see Figure 3.3(a) for a schematic display of the xy-projection of a counterexample, and Figure 3.3(b) for a cross section. Let the flat disk (red) be the first region, and the upper ridge (blue) the second region. While the boundaries of the two regions are semi-completely intervisible, even completely, there is a point in the interior of the disk that does not see any point on the ridge.



(a) (b)

Figure 3.3: The boundaries of two regions can be semi-completely intervisible, while the regions themselves are not semi-completely intervisible.

3.4.1 Aspect graph approach

To start, we compute whether every point in R_1 sees at least some point in R_2, i.e., we compute whether R_2 is weakly visible from every point in R_1. The other way around can be computed analogously. To obtain some more intuition for the problem at hand, we first describe a brute-force approach to compute whether R_2 is weakly visible from every point in R_1. In Section 3.4.2, we significantly improve on the $O(mn^8 \log n)$ brute-force time bound.

The aspect graph, first proposed by Koenderink and van Doorn [112], is an important structure for studying visibility in 3D. For our setting of a polyhedral terrain, this structure is defined as the graph in which the nodes represent the set of combinatorially different views of the terrain, and the arcs represent the transitions between these views due to continuous movement of the viewpoint. In this definition, a *view* refers to a view of the terrain such that an infinitesimal movement will yield a view of the object that has the same combinatorial structure. We do not use the aspect graph itself, but the subdivision of space it induces. This subdivision of $\mathbb{R}^3$ consists of cells in which all points in the same cell have the same view combinatorially. A shared boundary between two cells indicates a transition between the views of the corresponding cells. Although this is a slightly different structure than the original graph from [112], we call it an aspect graph in this chapter.

There are $O(n^3)$ transitions between different views of the terrain, and these can be subdivided into three types: $O(n^3)$ transitions are defined by three edges of the terrain, $O(n^2)$ transitions are defined by a vertex and an edge, and $O(n)$ are defined by a single triangle [20]. In Figure 3.4 these three types are illustrated.

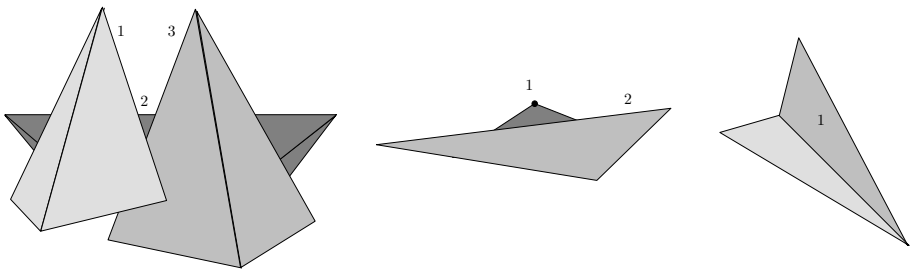


Figure 3.4: The three types of transitions between different views.

All of these transitions define either a plane or an algebraic surface of constant degree. If the viewpoint is on one side of such a plane or surface, the view is combinatorially different than if the viewpoint is on the other side. Because grazing contact of the line-of-sight with the terrain is allowed, the boundary between two cells has the same combinatorial view as the cell that 'sees the most' of the two. These $O(n^3)$ planes and surfaces and their intersections subdivide space into cells and thus the aspect graph has complexity $O(n^9)$. However, in [3], the complexity of this structure for a polyhedral terrain is proven to be $O(n^{8+\epsilon})$. If we compute the aspect graph of the terrain on R_1,

we get a partition of R_1 into a finite set of aspect graph cells inside which all points have the same view of the terrain. We compute the intersection of the visibility map and R_2 for one sample point in each such cell. This intersection is nonempty for each cell if and only if R_2 is weakly visible from R_1.

Because the $O(n^3)$ planes and surfaces partition a single triangle into $O(n^6)$ cells, we have $O(n^6)$ sample points for every triangle of R_1. For every sample point, we can compute the visibility map of the terrain in $O(n^2 \log n)$ worst-case time [110], so for a region of $O(m)$ triangles, the total time complexity will be $O(mn^8 \log n)$. In the next section, we improve on this bound by taking a different approach.

3.4.2 Using the edge visibility map

In this section, we take a conceptually different look at semi-complete intervisibility between two regions. In general, visibility and illumination problems are closely related; therefore, let us assume that every point in R_2 emits light, and then consider which points on the terrain, and in particular on R_1, are illuminated. These are exactly the points that see at least one point in R_2. In the terminology of Definition 3.2.1, we now have that R_1 weakly sees R_2 if and only if R_1 is completely illuminated. The following observation follows directly from the property that every vertical line intersects a terrain at most once.

Observation 3.4.1. *Let p be a point and R a region, both on a terrain T. If p is illuminated by some point in R, it is illuminated by a point on an edge of one of the triangles in R.*

This observation gives us a new way to test for semi-complete intervisibility between R_1 and R_2. First, for every illuminating edge of R_2, we compute the subdivision of T into illuminated and shadow points. Next, we overlay these $O(m)$ subdivisions. Finally, we determine whether or not R_1 is completely illuminated.

In Chapter 4, we show that an illuminating edge induces a *shadow map* of complexity $\Theta(n^2)$ onto another terrain edge and a shadow map of complexity $\Theta(n^4)$ onto a terrain triangle. The computation of these shadows takes $O(n^2 \log n)$ and $O(n^4 \log n)$ time, respectively. Note that the shadow map of a point light source is equivalent to the visibility map of that point acting as a viewpoint.

Fortunately, we do not need to overlay all the separate shadow maps explicitly, which would very time-consuming. The $O(n^4 \log n)$ time algorithm that computes the shadow map of a single illuminating edge e onto a triangle t of T first computes an arrangement on t that is induced by $O(n^2)$ critical surfaces and then labels each cell of this arrangement as being either illuminated or in the shadow (see Chapter 4. A multi-edge shadow map for $O(m)$ illuminating edges can be computed in the same way, with the difference that there now are $O(mn^2)$ critical surfaces that induce the arrangement, which thus has complexity $O(m^2n^4)$. This arrangement $\mathcal{A}$ can be computed in $O(m^2n^4)$ randomized expected time [143]. If we compute the $O(m)$ individual shadow maps in advance, in $O(mn^4 \log n)$ time, we can walk through each of these maps and simultaneously label the cells in $\mathcal{A}$ that are contained in an illuminated cell of one of

the shadow maps as illuminated. This labeling can be performed in $O(mn^4 \log n)$ time, since we walk through $O(m)$ arrangements of complexity $O(n^4)$. Thus, we can compute which parts of a terrain triangle t are illuminated by the edges of the triangles in R_1 in $O(m^2n^4)$ time, and for the entire terrain in $O(m^3n^4)$ time. Once we have this information, it is easy to determine whether R_2 is completely illuminated within the same time bound.

Theorem 3.4.1. *Let T be a terrain, and let R_1 and R_2 be two regions on T, with $O(m)$ triangles each. Semi-complete intervisibility of R_1 and R_2 can be determined in $O(m^3n^4)$ time.*

3.5 Partial intervisibility

Finally, we consider the problem of determining whether there exists a point in region R_1 that sees any point in R_2. The proof of the following lemma is straightforward and in fact very similar to the proof of Lemma 3.3.1.

Lemma 3.5.1. *Two regions R_1 and R_2 on a terrain are partially intervisible if and only if there exists a point p on an edge of a triangle in R_1 and a point q on an edge of a triangle in R_2 such that p sees q.*

Proof: The if-part of the proof is immediate from Definition 3.2.1.

For the only if-part, let p be a point in R_1 and let q be a point in R_2, such that p sees q. Let μ be the vertical plane through the line segment pq, and let $T_\mu = T \cap \mu$ be the cross section of T induced by μ. We only need to consider the visibility in T_μ to show that there exist a point v on an edge of R_1 and a point w on an edge of R_2 that see each other. Because T_μ is a lower-dimensional terrain, we can rotate the ray emanating from p and directed towards q upwards around p, until we hit a vertex v of T_μ, such that v and p are mutually visible. Note that v is a point on an edge of R_2; recall that R_2 consists of complete triangles of T. We repeat this process by rotating the ray from v towards p upwards, to find a vertex w of T_μ such that v sees w. Now we have that w is a point on an edge of R_1, and the fact that v and w see each other completes the proof. ∎

Using the same concept of illumination instead of visibility as in the previous section, this problem turns out to be quite easy to tackle. It is easy to verify that if the edges of R_1 act as light sources, and R_1 and R_2 are partially intervisible, then some edge in R_2 contains an illuminated point. As we showed in the previous section, the shadow map induced by an illuminating edge onto another edge has complexity $\Theta(n^2)$ and can be computed in $O(n^2 \log n)$ time. Since the shadow map is a one-dimensional structure, there is no need to actually overlay m shadow maps on every edge; we simply compute them sequentially and stop as soon as we find an illuminated point. In particular, we run the $O(n^2 \log n)$ time edge-to-edge shadow map algorithm of Chapter 4 for all $O(m^2)$ different edge pairs of one edge in R_1 and one in R_2. This gives us the following theorem:

Theorem 3.5.1. *Let T be a terrain, and let R_1 and R_2 be two regions on T, with $O(m)$ triangles each. Partial intervisibility of R_1 and R_2 can be determined in $O(m^2n^2 \log n)$ time.*

3.6 Concluding remarks

In this chapter, we considered three problems related to the intervisibility of two regions of at most m triangles on a terrain with n triangles. We first developed an algorithm that, for any $\epsilon > 0$, determines in $O(n^{1+\epsilon}m)$ time whether the two regions in a terrain are completely intervisible, whereas a brute-force algorithm would require $O(m^2n)$ time. The algorithm uses a partition tree and a cutting tree, both with associated structures, leading to a total of $O^*(mn)$ storage. Moreover, we proved that computing complete visibility between two regions is 3SUM-hard.

Secondly, we developed an algorithm to determine in $O(m^3n^4)$ time whether two regions in a terrain are semi-completely intervisible. The algorithm involves computing the shadow maps of the edges of the first region onto the second region, and checks whether the second region is completely illuminated (and vice versa). Finally, we presented a similar algorithm to determine partial region intervisibility, which runs in $O(m^2n^2 \log n)$ time.

It should be noted that within the same time bounds, the different kinds of intervisibility between a constant number of regions can be determined, because every extra region requires a constant number of repetitions of the algorithm which does not increase the execution time asymptotically.

A number of other intervisibility problems are still unexplored, especially those that concern partial visibility. A possible topic for future research is partial visibility from a watchtower of, for example, a path through a terrain or a collection of other watchtowers.

Chapter 4

Visibility maps of segments and triangles in 3D

The contents of this chapter will be published as:

E. Moet, C. Knauer, and M. van Kreveld.
Visibility maps of segments and triangles in 3D.
Computational Geometry: Theory and Applications,
Vol. 39, No. 3, pages 163–177, April 2008.

Abstract

Let T be a set of n triangles in three-dimensional space, let s be a line segment, and let t be a triangle, both disjoint from T. We consider the subdivision of T based on (in)visibility from s; this is the *visibility map* of the segment s with respect to T. The visibility map of the triangle t is defined analogously. We look at two different notions of visibility: strong (complete) visibility, and weak (partial) visibility. The trivial $\Omega(n^2)$ lower bound for the combinatorial complexity of the strong visibility map of both s and t is almost tight: we prove an $O(n^2\alpha(n))$ upper bound for both structures, where $\alpha(n)$ is the extremely slowly increasing inverse Ackermann function. Furthermore, we prove that the weak visibility map of s has complexity $\Theta(n^5)$, and the weak visibility map of t has complexity $\Theta(n^7)$. If T is a polyhedral terrain, the complexity of the weak visibility map is $\Omega(n^4)$ and $O(n^5)$, both for a segment and a triangle. We also present efficient algorithms to compute all discussed structures.

4.1 Introduction

Computations concerning visibility and the generation of shadows are important to obtain realistic images in computer graphics. Unfortunately, these computations can be very time-consuming. The *visibility map* of a point p with respect to a set T of n triangles in 3D, which we call a *3D scene*, is the set of maximally connected components on T that are visible from p:

Definition 4.1.1. *Let T be a 3D scene, and let p be a point in $\mathbb{R}^3$. We denote by $\mathrm{VP}(p,T)$ the visibility polyhedron of p with respect to T, i.e., the connected subset of $\mathbb{R}^3$ that contains all points that see p. The subdivision of the triangles of T induced by $\mathrm{VP}(p,T)$ is the* visibility map *of p and is denoted by $\mathrm{VM}(p,T)$.*

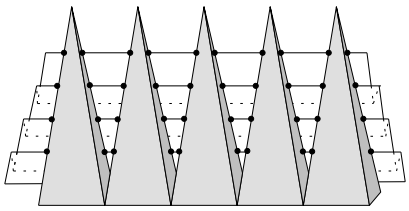


Figure 4.1: A visibility map of a point with $\Omega(n^2)$ vertices.

Through (perspective) projection onto a viewplane near p, the visibility map of the point p directly corresponds to a planar graph with $\Theta(n^2)$ vertices, edges, and faces [131, 151]; see Figure 4.1. In this chapter, we study the worst-case combinatorial complexity of the visibility map for a set that consists of *more than a single point*; in particular, we consider the cases where the view element is a line segment or a triangle.

Note that the above graph analogy is no longer valid in these cases. In the definitions below, s is a low-dimensional simplex, for example, a line segment or a triangle in $\mathbb{R}^3$.

Definition 4.1.2. *The* strong visibility polyhedron *of s, denoted by* $\mathrm{SVP}(s,T)$*, is the set of points in $\mathbb{R}^3$ that see every point in s. The subdivision of the triangles of T induced by* $\mathrm{SVP}(s,T)$ *is the* strong visibility map *of s; we denote this structure by* $\mathrm{SVM}(s,T)$.

Definition 4.1.3. *The* weak visibility polyhedron *of s, denoted by* $\mathrm{WVP}(s,T)$*, is the set of points in $\mathbb{R}^3$ that see at least one point of s. The subdivision of the triangles of T induced by* $\mathrm{WVP}(s,T)$ *is the* weak visibility map *of s; we denote this structure by* $\mathrm{WVM}(s,T)$.

In general, visibility polyhedra are connected sets, whereas visibility maps consist of multiple connected components. The *strong visibility map* of S identifies the maximally connected components on T in which the points see *every* point of s, whereas the other points of T do not see all of s, i.e., they see s partly or not at all. The *weak visibility map* separates the points that see at least one point in S, and the points that do not see any point in s. An alternative way of looking at the weak visibility map is to consider s to be a light source. Then the weak visibility map of s is equivalent to the set of maximally connected components on T that are illuminated; we will sometimes call the weak visibility map of s the *shadow map* of s.

With the results of Wang and Zhu [193], the weak visibility map of a triangle in a general 3D scene can be computed in $O(n^9)$ time and $O(n^7)$ space; to compute the same structure in a polyhedral terrain, the algorithm of [193] can be used to achieve a running time of $O(n^7)$ time and $O(n^5)$ storage. In this chapter, we improve significantly on these previous results and show that our bounds are almost tight by giving lower bound constructions. For a line (segment), a collection of general results can be found in [26]. Both of the above papers use the detection of transparent and opaque topology changes to compute the parts of a 3D scene that are weakly visible from a given view element; these topology changes are generally known as EEE events [91, 161].

A summary of the most important related results in computational geometry can be found in [151]. Recently, an $O(n^2 \log n)$ time algorithm was presented to compute the weak visibility map of an edge in a terrain restricted to another edge [46]. One of the main topics related to our study is the *aspect graph* [112]. This graph corresponds directly to the more frequently used partition of $\mathbb{R}^3$ into three-dimensional cells such that the visibility map with respect to the scene is combinatorially the same throughout each cell. The *critical points* where the view of the scene changes combinatorially lie exclusively on *critical surfaces* called reguli; see [91] and Section 4.2. The above mentioned spatial subdivision is induced by a set of $O(n^3)$ of these critical surfaces, and therefore has complexity $O(n^9)$. However, it has been shown that its complexity is almost a linear factor less for polyhedral terrains [3, 20].

Shadows are of major interest in computer graphics. The strong and weak visibility polyhedron of a view element are closely related to the *antiumbra* and *antipenumbra* of a light source, respectively. Many papers consider computing the shadow from a given point light source, either in image space, see [196], or in object space [42, 87]. The exact

shadow of a segment or area light source is considered too complex to compute exactly. This may be the reason that a majority of the studies until now have been incomplete; some address a particular type of 3D scene, e.g., a sequence of convex areal holes [186], others give algorithms that yield an approximation of the shadow [42, 46, 56, 99].

Table 4.1 gives a summary of our results; for completeness a previous result is included. All displayed results are worst-case combinatorial complexities, and $\alpha(n)$ is the extremely slowly increasing inverse Ackermann function. We also present algorithms to compute the discussed structures within almost matching time bounds, i.e., in most cases the computation time is only an $O(\log n)$ factor worse than the worst-case complexity.

	strong	weak (terrain)	weak (general)	
point	$\Theta(n^2)$			[131]
segment	$\Omega(n^2)$	$\Omega(n^4)$	$\Theta(n^5)$	Section 4.3 & 4.4
triangle	$O(n^2\alpha(n))$	$O(n^5)$	$\Theta(n^7)$	Section 4.5

Table 4.1: Combinatorial complexities of the visibility map in a 3D scene for various types of view elements under different models of visibility.

This chapter is organized as follows. In Section 4.2, we give several definitions and assumptions. We address the strong and the weak visibility map of an edge in Section 4.3 and 4.4, respectively. In Section 4.5, we study the complexity of the two types of visibility maps when the view element is a triangle, and we round up in Section 4.6 with the conclusions and a discussion of the obtained results.

4.2 Preliminaries

In this chapter, we consider sets of n triangles in $\mathbb{R}^3$ with pairwise disjoint interiors. We call such a set T a *3D scene*, or simply *scene*. Without loss of generality, we assume that the vertical direction is the z-direction. A special case of a scene is a *polyhedral terrain*, or simply *terrain*, in which T is a piecewise-linear continuous function defined over the triangles of a triangulation in the xy-plane. Given a scene T and two points $p, q \in \mathbb{R}^3$, we say that p *sees* q if the open line segment pq does not intersect the interior of any triangle in T, i.e., grazing contact of the line-of-sight with the edges of the triangles of T is permitted. In the case of terrains, two points can only be mutually visible if they both lie on or above the terrain. All our results, both the bounds on the worst-case complexities and the algorithms, hold when T contains degeneracies. However, to simplify argumentation, we only consider general position configurations in our proofs. This is no restriction; the worst-case complexities can be achieved by a 3D scene in general position, and in the presented algorithms, we can apply standard symbolic perturbation techniques [61].

To study visibility in $\mathbb{R}^3$, it is important to consider collections of lines that intersect three given lines. Such collections are called *reguli*. More formally, the regulus $\mathcal{R}(l_1, l_2, l_3)$ for three lines l_1, l_2, and l_3, is the set of all lines that intersect l_1, l_2, and l_3, in this (or the inverse) order. A regulus has the following properties [164, §3]:

1. it is one ruling of either a hyperbolic paraboloid, or a hyperboloid of one sheet;
2. it is an algebraic surface of constant degree;
3. the projection of its boundary onto the xy-plane is a set of algebraic curves of constant degree;
4. it can be partitioned into a constant number of xy-monotone parts.

Let $\ell(s)$ denote the supporting line of a line segment s. For three edges e, f, and g of T, all lines that intersect e, f, and g, in this (or the inverse) order, lie in a subset of $\mathcal{R}(\ell(e), \ell(f), \ell(g))$. This set consists of at most three connected components [36], and is denoted by $\mathcal{R}(e, f, g)$. The surface $\mathcal{R}(e, f, g)$ can be computed in $O(1)$ time [36]. The critical points where the view of the scene changes combinatorially lie exclusively on these reguli [91].

Throughout this chapter, we adopt an appropriate model of computation, i.e., we assume that computing various elementary operations on algebraic curves and surfaces of constant degree can be performed in constant time; for a discussion of computational real algebraic geometry, see for instance [101].

Let e be an edge of T, and let t be a triangle of T. In this chapter, we study the worst-case combinatorial complexities of the following four structures: $\mathrm{SVM}(e, T)$, $\mathrm{WVM}(e, T)$, $\mathrm{SVM}(t, T)$, and $\mathrm{WVM}(t, T)$. We assume that e or t is a feature of T, but this is no restriction; if an arbitrary segment or triangle (1- or 2-simplex) $s \subset \mathbb{R}^3$ does not intersect any triangle in T, and lies above T in case T is a polyhedral terrain, then we can transform s into a feature s' of a 3D scene T' in $O(n \log n)$ time in such a way that the visibility map of s in T is essentially the same as the visibility map of s' in T'. If T is a general 3D scene, this is straightforward. In the case of a triangular view element, we simply add this triangle to obtain the scene T'. For a line segment s, we create an additional skinny triangle that has s as an edge. The transformation is a little bit more involved when T is a polyhedral terrain. If the view element is a line segment s, which lies entirely above T, we build a near-vertical wall below s, and we triangulate the affected triangles of T to obtain the transformed terrain T'. If the view element is a triangle t that lies entirely above T, we consider the two types of visibility map separately. Because every point on T that lies below t is weakly visible from t, we can treat this part of the terrain separately and connect t downwards (again, near-vertical) with T to obtain T'. For strong visibility, we cannot do this, since it is not trivial to compute the strong visibility map of t for the part of T below t. In this case, we create a general 3D scene T', which is no longer a polyhedral terrain, that consists of t plus all the triangles of T. This is allowed, because the combinatorial complexity of the strong visibility map of a triangle is the same in a general 3D scene as it is in a polyhedral terrain (see Section 4.3).

As mentioned in the introduction, the strong and weak visibility polyhedron are closely related to the *antiumbra* and *antipenumbra* of a light source, respectively, which are well-known structures in computer graphics. More precisely, the antiumbra is the volume from which all points on the light source can be seen, and the antipenumbra is the volume from which some, but not all, of the light source can be seen [186]. In this setting, the strong visibility polyhedron is equal to the antiumbra, and the weak visibility polyhedron is equal to the closure of the union of the antiumbra and antipenumbra.

A trivial lower bound for the complexity of all four types of visibility maps is $\Omega(n^2)$, which is a lower bound for the complexity of the visibility map of a single point [131].

Definition 4.2.1. *[112] Let T be a 3D scene. The* aspect graph *induced by, and restricted to, T is the subdivision of the triangles of $\mathbb{R}^3$ into cells in which the perspective projection of T is combinatorially the same.*

The points in one cell of the aspect graph see exactly the same set of features of T; in particular, throughout each cell the simplex s is either visible or invisible. Thus, the complexity of the aspect graph restricted to T is an upper bound on both $\mathrm{SVM}(s,T)$ and $\mathrm{WVM}(s,T)$. On each of the n triangles of T, the aspect graph is defined by the arrangement of $O(n^3)$ critical surfaces, which has complexity $O(n^6)$, and thus the overall complexity is $O(n^7)$. Hence, the complexity of each of the above defined visibility maps is $\Omega(n^2)$ and $O(n^7)$; these bounds also hold for visibility maps that we do not discuss here, e.g., the shadow map of a polyhedral light source.

4.3 The strong visibility map of an edge

The 2D counterpart of the strong visibility polyhedron of an edge in a 3D scene is the strong (or alternatively, complete) segment visibility polygon $\mathrm{SVP}(e,P)$ for an edge e in a simple polygon P, where P may have holes. This strucure has complexity $\Theta(n)$ [10, p.837]. In fact, the strong visibility polygon of a segment pq within P is equal to the visibility polygon of the point p *within* the visibility polygon of the point q in P, i.e., $\mathrm{SVP}(pq,P) = \mathrm{VP}(p,\mathrm{VP}(q,P))$.

Before we prove a lower and upper bound on the complexity of $\mathrm{SVM}(e)$, we prove that the analog of the above holds in three dimensions as well. By Δ_{sp}, we denote the triangle with base s and apex p.

Lemma 4.3.1. *Let T be a 3D scene, and let $e = vw$ be an edge of T. The strong visibility polyhedron* $\mathrm{SVP}(e,T)$ *is equal to* $\mathrm{VP}(v,\mathrm{VP}(w,T))$.

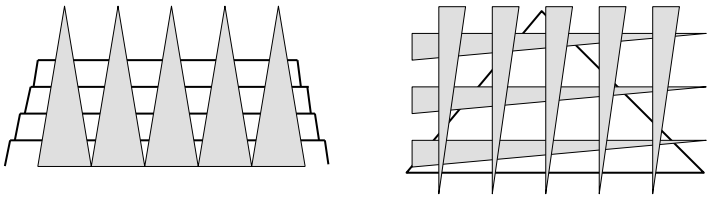
Proof: First, we show that if $p \in \mathrm{SVP}(e,T)$ then $p \in \mathrm{VP}(v,\mathrm{VP}(w,T))$. The triangle Δ_{ep} does not intersect any triangle of T. In particular, the line segment vp does not intersect T, so $p \in \mathrm{VP}(v,T)$. Because the whole of Δ_{ep} is free from intersections with T, v sees all points on the line segment wp as well. This means that the line segment $wp \subset \mathrm{VP}(v,T)$ and thus that w sees p within $\mathrm{VP}(v,T)$.

Second, we show that if $p \in \mathrm{VP}(v,\mathrm{VP}(w,T))$ then $p \in \mathrm{SVP}(e,T)$. We have that p sees both vertices v and w, which means the line segments vp and wp are not intersected

by T. Furthermore, both these line segments, and the edge e, lie in $\mathrm{VP}(v, T)$, which is a star-shaped polyhedron. This immediately implies that the triangle Δ_{ep} completely lies in $\mathrm{VP}(v, T)$. This means that Δ_{ep} is free from intersections with T, and thus $p \in \mathrm{SVP}(e, T)$. ∎

Unfortunately, this analysis does not provide us with a good upper bound on the complexity of $\mathrm{SVP}(e, T)$, since the visibility polyhedron of a point in a 3D scene already has complexity $\Theta(n^2)$ in the worst case, and the visibility polyhedron of a point in a scene with $O(n^2)$ triangles has complexity $O(n^4)$. Therefore, we take a different approach.

In the remainder of this section, we are given a 3D scene T with n triangles and an edge e of T, and we aim to bound the complexity of the strong visibility map $\mathrm{SVM}(e, T)$ of e on T. A trivial lower bound for $\mathrm{SVM}(e, T)$ is $\Omega(n^2)$; see Section 4.2 and Figure 4.2.



(a) (b)

Figure 4.2: (a) A terrain that induces a strong edge visibility map of total complexity $\Omega(n^2)$. (b) A single triangle in a 3D scene (no polyhedral terrain) with a strong edge visibility map of $\Omega(n^2)$ on it. In both figures, the shaded triangles in the front induce edges and vertices on the white triangle(s) in the background.

Lemma 4.3.2. *The strong visibility polyhedron* $\mathrm{SVP}(e, T)$ *of an edge* e *in a 3D scene* T *is connected.*

Proof: Let p and q be two points in $\mathbb{R}^3$ that strongly see e; this implies that the interiors of the triangles Δ_{ep} and Δ_{eq} are free from intersections with the interiors of the triangles in T. Let x be any point on e. The two line segments px and qx are free from intersections with T as well. Furthermore, for every point y on either px or qx, the triangle Δ_{ey} does not intersect T, and thus y is in $\mathrm{SVP}(e, T)$. Therefore, $\mathrm{SVP}(e, T)$ is connected. ∎

Since the triangles of T already induce $\mathrm{SVP}(e, T)$, the complexity of $\mathrm{SVM}(e, T)$ is at most the complexity of $\mathrm{SVP}(e, T)$. We analyze the complexity of $\mathrm{SVP}(e, T)$ to obtain an almost tight $O(n^2\alpha(n))$ upper bound on the complexity of $\mathrm{SVM}(e, T)$.

Definition 4.3.1. *Let* p *be a point, and let* s *be a line segment, both in* $\mathbb{R}^3$*. We define* $\mathcal{P}(p, s)$ *to be the set of points* q *for which there exists a ray starting at* p *that first passes through* s *and then through* q*. Similarly,* $\mathcal{P}(s, p)$ *is the set of points* q *for which there exists a ray starting at a point on* s *that first passes through* p *and then through* q*.*

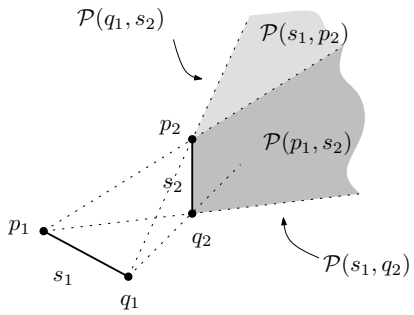


Figure 4.3: Illustration to Definition 4.3.1.

For two line segments $s_1 = p_1q_1$ and $s_2 = p_2q_2$, the (unbounded) region enclosed by $\mathcal{P}(s_1, p_2)$, $\mathcal{P}(p_1, s_2)$, $\mathcal{P}(q_1, s_2)$, and $\mathcal{P}(s_1, q_2)$ contains exactly the points p for which the triangle with base s_1 and apex p intersects the segment s_2; see Figure 4.3. For our purposes, if v is a vertex and e an edge of T, where v and e are not incident to each other, we represent the unbounded object $\mathcal{P}(e, v)$ by a large triangle, by clipping it with a vertical plane that lies far enough such that we do not lose any features of $\mathrm{SVP}(e, T)$. Furthermore, we clip $\mathcal{P}(v, e)$ in the same way and represent it by a finite-sized quadrilateral. From now on, we consider a triangle to be a degenerate quadrilateral, in which one side has zero length.

Definition 4.3.2. *Given a 3D scene T and an edge $e = vw$ of T, we define $S(e, T)$ to be the following set of (convex) quadrilaterals:*

$$S(e, T) = T \cup \mathcal{E} \cup \mathcal{V} \cup \mathcal{W},$$

where $\mathcal{E} = \{\mathcal{P}(e, u) \mid u$ vertex of $T\}$, $\mathcal{V} = \{\mathcal{P}(v, f) \mid f$ edge of $T\}$, and $\mathcal{W} = \{\mathcal{P}(w, f) \mid f$ edge of $T\}$.

If we split each quadrilateral in this set into a pair of triangles, $\mathrm{SVP}(e, T)$ is a single cell of the arrangement in 3D induced by $S(e, T)$ and consequently has complexity $O(n^2 \log n)$ [136]. In the lemma below, we improve on this result by counting the number of features of this single cell on the quadrilaterals of $S(e, T)$ separately.

Lemma 4.3.3. *Let T be a 3D scene, and let e be an edge of T.* $\mathrm{SVP}(e, T)$ *has complexity* $O(n^2\alpha(n))$.

Proof: The combinatorial complexity of $\mathrm{SVP}(e, T)$ is the complexity of the cell in question of the arrangement induced by $S(e, T)$. This complexity is the number of faces, which are part of the quadrilaterals of $S(e, T)$, plus the number of edges and vertices, which are intersections between, respectively, two or three of these quadrilaterals. We prove that this number is $O(n^2\alpha(n))$ by showing that it is $O(n\alpha(n))$ on each triangle in $\mathcal{E} \cup \mathcal{V} \cup \mathcal{W}$. Each feature on a (scene) triangle in T can be charged to a feature on one of the triangles of $\mathcal{E} \cup \mathcal{V} \cup \mathcal{W}$, since the triangles in T have disjoint interiors.

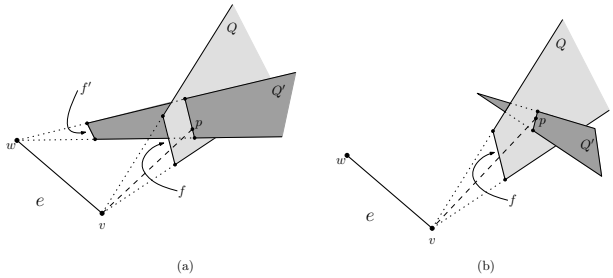


Figure 4.4: Illustration of the proof of Lemma 4.3.3.

Let Q be a quadrilateral in $\mathcal{V}$; the cases for $\mathcal{W}$ and $\mathcal{E}$ are analogous. The remaining quadrilaterals from $S(e,T)$ induce a two-dimensional arrangement $\mathcal{A}_Q$ of line segments on Q. We count the number of features of $\mathcal{A}_Q$ that contribute to the complexity of $\mathrm{SVP}(e,T)$. Let f be the edge of T such that $Q = \mathcal{P}(v,f)$, and let p be a point in $\mathrm{SVP}(e,T) \cap Q$. The point p lies on a line segment that is the intersection of Q with another quadrangle Q' from $S(e,T)$. In Figure 4.4, we show the cases where (a) Q' is from $\mathcal{W}$, and (b) Q' is a (scene) triangle from T. The third case is that Q' is from $\mathcal{E}$. In each case, let s be the line segment that connects the point p with v. Because p is in $\mathrm{SVP}(e,T)$, s intersects exactly one feature of T, namely the edge f.

No other quadrilaterals from $S(e,T)$ can intersect s, otherwise p would not see e completely. In other words, every point in $\mathcal{A}_Q$ for which the line segment that connects it with v intersects no other quadrilateral from $S(e,T)$ is in $\mathrm{SVP}(e,T) \cap Q$. If we consider the direction towards v as the upward direction, the boundary of $\mathrm{SVP}(e,T)$ on Q directly corresponds to the upper envelope of a set of $O(n)$ Jordan curves that pairwise intersect at most once; see Figure 4.5. The set of all points on Q that see e completely together form a face of $\mathrm{SVP}(e,T)$ that lies on Q (in fact, it is the only face of $\mathrm{SVP}(e,T)$ on Q), which is a connected region in $\mathcal{A}_Q$. Since the complexity of an upper envelope of such segments has complexity $O(n\alpha(n))$ [173], the number of edges and vertices of $\mathrm{SVP}(e,T)$ that lie on Q is $O(n\alpha(n))$. Hence, we have that $\mathrm{SVP}(e,T) \cap Q$ has complexity $O(n\alpha(n))$ for each quadrilateral Q from $\mathcal{E} \cup \mathcal{V} \cup \mathcal{W}$, and the total complexity of $\mathrm{SVP}(e,T)$ is $O(n^2\alpha(n))$.

■

In conclusion, we have the following theorem.

Theorem 4.3.1. *Let T be a 3D scene with n triangles, and let e be an edge of T. The strong visibility map* $\mathrm{SVM}(e,T)$ *has complexity* $\Omega(n^2)$ *and* $O(n^2\alpha(n))$.

The proof of Lemma 4.3.3 gives us a way to efficiently compute the strong visibility map:

Algorithm COMPUTE_STRONG_VM :
INPUT: A set T of n triangles in $\mathbb{R}^3$ and an edge e of T.

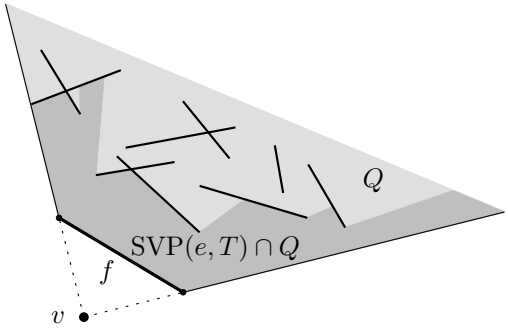


Figure 4.5: Illustration of the proof of Lemma 4.3.3.

OUTPUT: SVM(e, T)

1. Generate the set of quadrilaterals $S(e, T)$ as defined in Definition 4.3.2.

2. For each quadrilateral Q in $\mathcal{E} \cup \mathcal{V} \cup \mathcal{W}$:

   (a) Compute the intersections of Q with the other quadrilaterals in $S(e, T)$.

   (b) Compute the upper envelope of these line segments; this is the boundary of SVP(e, T) on Q.

   (c) Identify those features of the envelope that are induced by a triangle of T and store this information with that particular scene triangle.

3. For each triangle of T: compute the visible connected components from the information computed above. Output this set of connected components; this is SVM(e, T).

The first step trivially takes $O(n)$ time, and step 2(a) $O(n^2)$ time. In step 2(b), for each of the $O(n)$ quadrilaterals from $\mathcal{E} \cup \mathcal{V} \cup \mathcal{W}$, we use the optimal algorithm of Hershberger [103] to compute the boundary of SVP(e, T) in $O(n \log n)$ time, thus taking $O(n^2 \log n)$ time overall. In step 2(c), we check the features that we computed in step 2(b) for intersection with triangles from T, which can trivially be done in $O(n\alpha(n))$ time per quadrilateral in $S(e, T)$, so this step takes $O(n^2\alpha(n))$ time in total. In step 3, we glue the connected components together; this can e.g. be done by a straightforward sweepline algorithm, which takes $O(n^2\alpha(n) \log n)$.

Theorem 4.3.2. *Let T be a 3D scene with n triangles, and let e be an edge of T. The strong visibility map of e on T can be computed in $O(n^2\alpha(n) \log n)$ time.*

4.4 The weak visibility map of an edge

4.4.1 Lower bounds

The weak visibility region of an edge in a polygon with holes can have complexity $\Omega(n^4)$, as was shown by Suri and O'Rourke [183]. In their construction, an edge light source and two rows of segments with suitably placed gaps A and B produce a quadratic number of narrow light cones that all mutually intersect. These intersections of cones result in $\Omega(n^4)$ illuminated vertices of the visibility region; see Figure 4.6 for a schematic illustration of this construction.

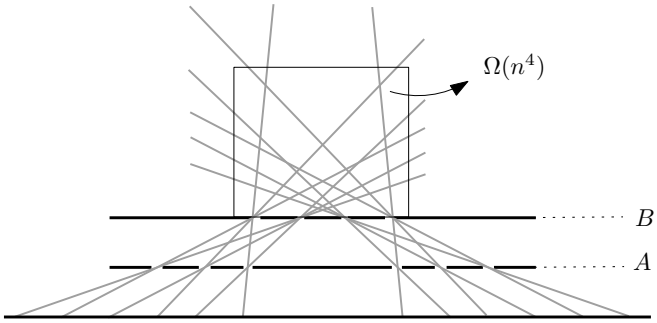


Figure 4.6: A schematic illustration of the lower bound construction for the weak visibility region of an edge in a polygon with holes [183].

We can create this same construction with a polyhedral terrain, where the 2D situation described above is replicated on the xy-plane, such that a single triangle contains all the $\Omega(n^4)$ intersections of the light cones. Thus, we have the following lemma.

Lemma 4.4.1. *The weak visibility map of an edge in a polyhedral terrain with n triangles can have $\Omega(n^4)$ vertices.*

For general scenes, we can improve this lower bound. We construct a scene such that every one of the $\Omega(n^2)$ light cones is unbounded in the z-direction, i.e., we have $\Omega(n^2)$ illuminated vertical planes, and each of the $\Omega(n^4)$ intersections of two cones is an illuminated vertical line. By placing a linear number of triangles vertically above each other, we can replicate the triangle that contains $\Omega(n^4)$ vertices of the visibility map $\Omega(n)$ times. Note that the triangles in this stack should be spaced in such a way that the light can extend through to the back of the stack. A schematic display of the lower bound construction for general 3D scenes is shown in Figure 4.7. Due to the vertically stacked triangles in C, this construction cannot be built as a polyhedral terrain.

Lemma 4.4.2. *The weak visibility map of an edge in a 3D scene with n triangles can have $\Omega(n^5)$ vertices.*

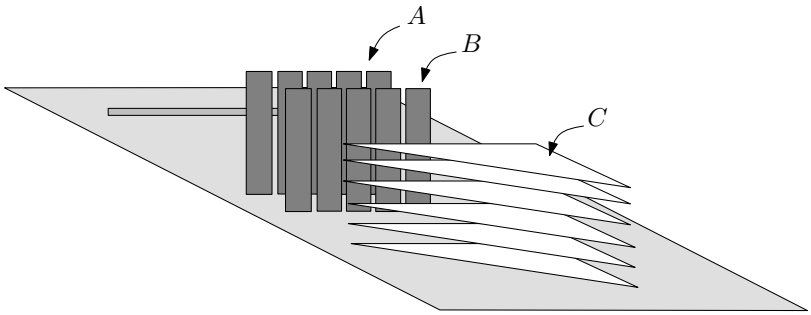


Figure 4.7: A schematic display of a 3D scene in which the weak visibility map of an edge has complexity $\Omega(n^5)$.

4.4.2 Upper bounds

Throughout this section, we are given a 3D scene T with n triangles and a designated edge e of T that acts as a light source. We prove upper bounds on the worst-case complexity of the shadow that e induces, together with the scene triangles, onto different features of T. In Section 4.4.2 we prove a bound of $\Theta(n^2)$ for the complexity of the shadow on an edge of T, and in Section 4.4.2 we prove $\Theta(n^4)$ for the complexity of the shadow on a triangle of T. Note that this last result, together with Lemma 4.4.2, implies that the visibility map of an edge in a general 3D scene has complexity $\Theta(n^5)$.

First, we look at the shadow that an edge light source e induces on a single point p on a 3D scene, which is equivalent to asking whether p is illuminated or not. Let Δ be the triangle that has p as its apex and the edge e as its base. We compute the set S of line segments that are intersections of the $O(n)$ triangles of T with Δ. This can be done in $O(n)$ time. Then, we project S onto e with respect to p, and compute whether these projections, which are intervals on e, cover e completely. We can do this by using an $O(n \log n)$ sweepline algorithm. It is easy to see that p is not illuminated by e if and only if e is completely covered by the projections of S, which can be checked while we sweep the intervals.

The above bound on deciding whether or not a point is illuminated is in fact tight. The $\Omega(n \log n)$ lower bound follows from an adaptation of the lower bound proof of Klee's Measure Problem: given a set of n (possibly overlapping) intervals $[a_i, b_i]$, compute the measure of their union. This problem is known to have a lower bound of $\Omega(n \log n)$ in the algebraic decision tree model [78], and the proof can easily be adjusted to provide an $\Omega(n \log n)$ lower bound for this illumination problem.

Complexity on an edge

By the lower bound construction in [183], the shadow map restricted to an edge can have complexity $\Omega(n^2)$, since there can be a quadratic number of light cones that intersect the edge in distinct places. The aspect graph gives us a trivial upper bound on the

shadow of $O(n^3)$, because that is the maximum number of different views of the light source e seen from f. We show that the $\Omega(n^2)$ lower bound is tight.

Lemma 4.4.3. *Let T be a 3D scene with n triangles, and let e and f be two edges of T. If e acts as a light source, then the shadow that T casts on f has complexity $O(n^2)$.*

Proof: If no point on f is illuminated, then the shadow map on f has complexity $O(1)$ and the lemma is trivially true. Otherwise, let p be an arbitrary illuminated point on f, and let q be a point on e that illuminates p; the open line segment pq does not intersect T. Now, we start moving p over f, in either of the two directions. We can keep moving p either until we reach the end of f, see Figure 4.8(a), or until the line segment pq intersects an edge f' of T. By construction, the point p' we determine in this way is illuminated, namely by q. Next, we move the point p' over f in the same direction as we moved p, while maintaining contact with f'. This corresponds to sweeping the plane defined by p' and f' over e. We keep moving until we reach the end of f, which is similar to the case in Figure 4.8(a), we reach the end of either e or f' (see Figure 4.8(b)), or touch a second edge g of T (see Figure 4.8(c)). In each case, we determine a point p'' on f that is illuminated by a point q'' on f.

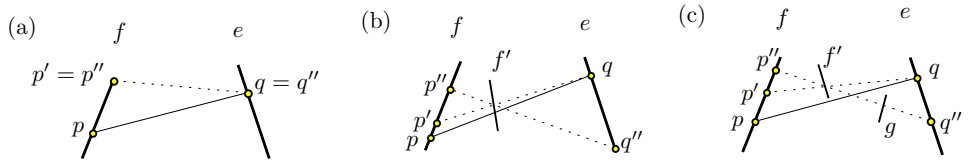


Figure 4.8: Three cases that arise in the proof of Lemma 4.4.3.

Observe that p'' is (a) an endpoint of f, (b) an extremal point on f of one of the connected components of the regulus $\mathcal{R}(e, f', f)$ for some edge f' of T, or (c) a point of intersection of f with the regulus $\mathcal{R}(e, f', g)$ or the regulus $\mathcal{R}(e, g, f')$, where f' and g are edges of T. Analogously, we can find a similar illuminated point if we start by moving p in the opposite direction. In the remainder of the proof, we bound the number of such points on f.

For every pair of edges (f', g) of T, we consider the regulus $\mathcal{R}(e, f', g)$. If this set exists, then it intersects the edge f at most in a constant number of points (see Section 4.2). Consider the arrangement $\mathcal{A}(f)$ on f that is induced by the intersections of the reguli $\mathcal{R}(e, f', g)$ with e, for all ordered pairs (f', g) of edges of T, with $f' \neq g$. Since there are $O(n^2)$ such reguli, the arrangement $\mathcal{A}(f)$ has complexity $O(n^2)$. By the discussion above, every interval $i \in \mathcal{A}(f)$ is completely illuminated or completely dark. Thus, $O(n^2)$ is an upper bound on the complexity of the shadow map of e on f. ∎

Complexity on a triangle

By the lower bound construction in [183], the shadow on a triangle t of a 3D scene can have complexity $\Omega(n^4)$. The aspect graph gives us a trivial upper bound on the shadow

of $O(n^6)$. Below, we prove that the lower bound is tight.

Lemma 4.4.4. *Let T be a 3D scene with n triangles, let e be an edge of T, and let t be a triangle of T. If e acts as a light source, then the shadow that T casts on t has complexity $O(n^4)$.*

Proof: If no point on t is illuminated, then the shadow has complexity $O(1)$ and the lemma is trivially true.

Otherwise, let p be an arbitrary illuminated point on t, and let q be a point on e that illuminates p; the open line segment pq does not intersect T. Now we choose any arbitrary straight line segment s through p that divides t in two. By the argumentation in the proof of Lemma 4.4.3, we can find an illuminated interval on s, not necessarily maximal, that contains p and is bounded on both sides by either an endpoint of s, or by a point that is the intersection of s with a line that goes through e as well as two edges f and g of T. Because s was chosen arbitrarily, we can find a two-dimensional illuminated region on t that contains p and is bounded by curves that are intersections of T with one or more reguli $R(e, f_i, g_j)$, where f_i and g_j are edges of T.

Consider the arrangement $\mathcal{A}(t)$ on t that is induced by the curves that are intersections of all possible reguli $\mathcal{R}(e, f, g)$ with t, for all ordered pairs (f, g) of edges of T, with $f \neq g$. Because there are $O(\binom{n}{2})$ edge pairs (f, g), there are $O(n^2)$ such curves, and this arrangement on t has complexity $O(n^4)$. By the discussion above, every cell $c \in \mathcal{A}(t)$ that contains at least one illuminated point, must be completely illuminated. Every cell $c' \in \mathcal{A}(e)$ that contains at least one shadow point, by the same arguments, must lie completely in the dark. Thus, $O(n^4)$ is an upper bound on the complexity of the shadow on t. ∎

Putting Lemmas 4.4.1, 4.4.2 and 4.4.4 together yields the following theorem.

Theorem 4.4.1. *Let T be a 3D scene with n triangles, and let e be an edge T. Then* $\mathrm{WVM}(e, T)$ *has complexity $\Theta(n^5)$ in the worst case. If T is a polyhedral terrain, then the complexity of* $\mathrm{WVM}(e, T)$ *is $\Omega(n^4)$ and $O(n^5)$.*

4.4.3 Algorithm

To compute the shadow of T that is cast onto an edge f by the illuminating edge e, we use a sweep algorithm. Our algorithm is very similar to the one described in [46]; however, we sweep the *illuminated* edge instead of the *illuminating* one. The reason for this adjustment is that, in this way, we immediately obtain the subdivision of f into illuminated and dark points, i.e., the shadow map of e on f.

First, we compute the set R of all reguli $R(e, f', g)$ for all ordered edge pairs (f', g) of T; there are $O(n^2)$ such reguli. Then we compute $\mathcal{A}(R, f)$, the arrangement on f induced by the intersections of the reguli in R with f. For one of the endpoints of f, say p_0, we compute the intersections of the triangles of T with the triangle Δ_{ep_0}. We project these intersections onto e; we now have a set S of $O(n)$ intervals on e, in these intervals either every point illuminates p_0 or none of them does. Now we move the point p_t, which initially is the point p_0, over the edge f, which corresponds to sweeping

the triangle Δ_{ep_t} through T. In the meantime, we maintain the set S of (projected) intervals on e. The *event points* of the sweep are the points where (i) a new interval (either illuminating or not) appears, (ii) an existing interval disappears, or (iii) two endpoints of different intervals change order. Because the intervals correspond directly to (sets of) triangles of T, it is easy to see that the event points are exactly the interval boundaries of $\mathcal{A}(R, f)$. An interval i of $\mathcal{A}(R, f)$ is illuminated if there is a point on e that illuminates any point $p \in i$, and is in the shadow otherwise.

Computing $\mathcal{A}(R, f)$ takes $O(n^2 \log n)$ time, as we compute and sort the intersections of a quadratic number of reguli with an edge, and every regulus intersects an edge a constant number of times. Computing the set S of intervals on e that illuminate p_0 takes $O(n \log n)$ time, as we showed at the beginning of Section 4.4.2. The number of event points we encounter during the sweep is $O(n^2)$ [46] and at every event point we need to update the status structure in a constant number of places, since there are exactly two edges of T associated with each event point. With an appropriate data structure, e.g., an augmented binary search tree, this step, and thus the complete algorithm, takes $O(n^2 \log n)$ time.

Lemma 4.4.5. *Let T be a 3D scene with n triangles, and let e and f be two edges of T. We can compute the part of* $\mathrm{WVM}(e, T)$ *restricted to f in $O(n^2 \log n)$ time.*

The shadow of T cast onto a triangle t of T by the illuminating edge e can be computed by a sweep algorithm as well. First, we need an extra lemma. Since two algebraic curves of constant degree in the plane have at most a constant number of intersections, it is easy to verify the lemma below (the proof is basically similar to the proof of Lemma 4.4.3 and the analysis in [46]).

Lemma 4.4.6. *Let e be an edge of a 3D scene T with n triangles, and let c be an algebraic curve of constant degree on a triangle t of T. If e acts as a light source, then the shadow that T casts on c has complexity $O(n^2)$ and can be computed in $O(n^2 \log n)$ time.*

The first step in computing the shadow map of an edge e on a triangle t is to generate the set $S(t)$ of curves of intersection of the reguli $\mathcal{R}(e, f, g)$ with t, for all ordered edge pairs (f, g) of T. Next, we compute the subdivision of t induced by $S(t)$, denoted by $\mathcal{A}(S(t))$. Let K be the complexity of this arrangement, which is $O(n^4)$ in the worst case. The arrangement $\mathcal{A}(S(t))$ can be computed in $O(n^2 \log n + K \log n)$ time [98]. We execute the $O(n^2 \log n)$ algorithm that computes the shadow map on an algebraic curve *twice* for every curve c that appears in $\mathcal{A}(S(t))$; we compute curves c' and c'' that, symbolically, lie infinitesimally close to c on either side. By computing the shadow maps on c' and c'' we can determine whether the adjacent cells of $\mathcal{A}(S(t))$ are **illuminated** or **dark**. If, during the construction of $\mathcal{A}(S(t))$, we store with every edge from which curve it originates, this cell labeling can be performed in $O(K \log n)$ time. Although the output-sensitive construction of $\mathcal{A}(S(t))$ will usually mean a more efficient algorithm in practice, it does not imply that we can compute the visibility map in an output-sensitive way, since not every regulus necessarily separates visible and invisible points. Thus, the running time of the above algorithm is $O(n^4 \log n)$ in the worst case.

Lemma 4.4.7. *Let T be a 3D scene with n triangles, let e be an edge of T, and let t be a triangle of T. We can compute the part of the weak visibility map of e on t in $O(n^4 \log n)$ time.*

Now it is trivial to extend the above procedure to compute the complete weak visibility map of e on T.

Algorithm COMPUTE_WEAK_VM :
INPUT: An edge e and a 3D scene T.
OUTPUT: $\mathrm{WVM}(e, T)$

1. For every triangle t of T, compute the shadow induced by e on t in the way described above.

2. Return the set of all illuminated connected components, which is $\mathrm{WVM}(e, T)$.

We now have obtained the following result.

Theorem 4.4.2. *Let T be a 3D scene with n triangles, and let e be an edge of T. The weak visibility map $\mathrm{WVM}(e, T)$ can be computed in $O(n^5 \log n)$ time.*

4.5 The triangle visibility map

We also consider the problem where the light source is neither a point nor a line segment, but a triangle in 3D. Using the techniques and ideas of the previous two sections, we show bounds on the worst-case combinatorial complexity of the two types of visibility maps for a triangle in 3D.

4.5.1 The strong visibility map of a triangle

The strong visibility map of a triangle t in a 3D scene T is the subdivision of the triangles of T into points that completely see t and points that do not. Like in Section 4.3, we bound the combinatorial complexity of the strong visibility polyhedron $\mathrm{SVP}(t, T)$, which also gives us an upper bound on the complexity of the strong visibility map $\mathrm{SVM}(t, T)$.

Let e_1, e_2, and e_3 be the three edges of t. We define $S(t, T)$ to be the set of quadrilaterals $\bigcup_{i=1..3} S(e_i, T)$, with $S(e_i, T)$ as in Definition 4.3.2. By the results in [136] and a similar argumentation as in the proof of Lemma 4.3.3, $\mathrm{SVP}(t, T)$ is a single cell in the three-dimensional arrangement induced by $S(t, T)$ with complexity $O(n^2\alpha(n))$. The algorithm to compute $\mathrm{SVM}(t, T)$ is similar to the one described in Section 4.3, which gives us the result below.

Theorem 4.5.1. *Let T be a 3D scene with n triangles, and let t be a triangle of T. Then $\mathrm{SVP}(t, T)$ and $\mathrm{SVM}(t, T)$ have complexity $\Omega(n^2)$ and $O(n^2\alpha(n))$ and can be computed in $O(n^2\alpha(n) \log n)$ time.*

4.5.2 The weak visibility map of a triangle

As we did in Section 4.4, we distinguish the two cases where the scene T is a terrain, and where T is a general 3D scene.

Polyhedral terrains

The lower bound construction where the visibility map of an edge has $\Omega(n^4)$ vertices (see Figure 4.6) trivially yields the same lower bound for a triangle light source.

The $O(n^5)$ upper bound on the complexity of the weak visibility map of an edge is also an upper bound for the weak visibility map of a triangle t. To verify this, suppose there is a point p in the weak visibility map of $\mathrm{WVM}(t)$, and let q be a point on t that sees p. Then we can rotate the line-of-sight pq vertically upwards around p, until we obtain a line-of-sight pq' that is free from intersections with T, such that the point q' lies on one of the edges of t. Hence, $\mathrm{WVM}(t) \subseteq \bigcup_{i=1..3} \mathrm{WVM}(e_i)$. Trivially, it also holds that $\mathrm{WVM}(t) \supseteq \bigcup_{i=1..3} \mathrm{WVM}(e_i)$. Because on every triangle of T, the complexity of $\bigcup_{i=1..3} \mathrm{WVM}(e_i)$ is bounded by the complexity of the arrangement induced by three sets of $O(n^2)$ reguli, $\mathrm{WVM}(t, T)$ is $\Theta(n^4)$ on a single triangle and $O(n^5)$ overall.

Theorem 4.5.2. *Let T be a polyhedral terrain, and let t be a triangle of T. Then $\mathrm{WVM}(t, T)$ has complexity $\Omega(n^4)$ and $O(n^5)$.*

General 3D scenes

Now, we consider the complexity of the weak visibility map of a triangle t on a general 3D scene T. Recall from Definition 4.2.1 in Section 4.2 that the aspect graph is the subdivision of space into connected regions where the view is combinatorially the same. In other words, all points in a single cell of the aspect see exactly the same features of T. Obviously, in every cell the triangle t is either visible or invisible. Thus, the complexity of the aspect graph *restricted to the triangles* of T is an upper bound on $\mathrm{WVM}(t, T)$. The complete aspect graph, i.e., the subdivision of $\mathbb{R}^3$, is defined by $O(n^3)$ critical surfaces [20]. For every triangle, this gives us a set of $O(n^3)$ critical curves, which in turn induces an arrangement of complexity $O(n^6)$ on a single triangle, and thus $O(n^7)$ on T. Hence, we have the following lemma.

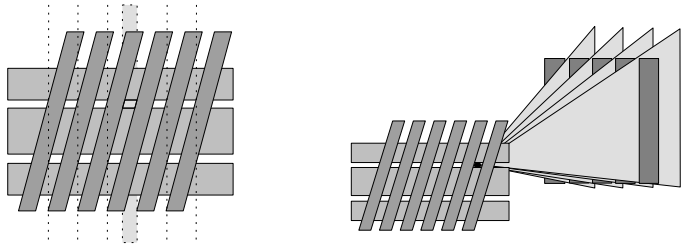
Lemma 4.5.1. *Let T be a 3D scene and let t be a triangle of T. Then $\mathrm{WVM}(t, T)$ has complexity $O(n^7)$.*

In the remainder of this section, we show that $O(n^7)$ is tight, by extending the $\Omega(n^5)$ lower bound construction of Figure 4.7. The basic idea of the extension is the following: first, we 'replace' A by a row giving $\Omega(n^2)$ gaps, which produces $\Omega(n^3)$ illuminated vertical planes, and thus we get $\Omega(n^6)$ illuminating vertical lines that each intersect $\Omega(n)$ triangles. In this way, we create $\Omega(n^7)$ vertices. We describe this construction in detail.

In order to construct a row with $\Omega(n^2)$ gaps that let the light that is emitted by t pass through without using more than $O(n)$ triangles, we first make the following observation. Each of the illuminated vertical planes of the lower bound of Figure 4.7

is uniquely determined by two gaps: one in A and one in B. Besides the obvious restriction that the light source should be located on the opposite side of A than B, these illuminated planes do not depend on the location of the light source. Because we now have an extra degree of freedom, we have the opportunity to make $\Omega(n^2)$ gaps by building a grid G instead of the row A. This idea is illustrated in Figure 4.9(a). The important fact here is that, if the holes in the grid degenerate to points, every hole has a unique x-coordinate.

If we allow enough distance between G and B, the holes in G basically act as point light sources, emitting cones of light. Now, just like in the old construction, each of the holes in G interacts with each of the gaps in B, producing $\Omega(n^3)$ illuminated vertical planes in total; see Figure 4.9(b).



(a) (b)

Figure 4.9: (a) A vertical grid with $\Omega(n^2)$ holes that allow light to pass through. (b) Every combination of a hole in the grid and a gap between two vertical slabs induces $\Omega(n)$ illuminated planes. (c) The total lower bound construction. The copy of G and B that induces another $\Omega(n^3)$ illuminated planes is not shown.

The total construction is illustrated schematically in Figure 4.10. First, we have the illuminating triangle t, followed by the grid G, then the row of vertical slabs B, and finally the set C of vertically stacked triangles. To ensure that we indeed get $\Omega(n^6)$ illuminated vertical lines, we place a copy of G and B on the xy-plane such that they generate $\Omega(n^3)$ illuminated vertical planes from a different direction; the light source must be large enough to provide light for both constructions. This gives us another set of $\Omega(n^3)$ illuminated vertical planes, intersecting the $\Omega(n^3)$ planes that are produced by the first copy.

Note that we have to make sure that all the vertices we create on the white triangles of Figure 4.10 are indeed vertices of the weak visibility map of t. To achieve this, we add a constant number of very large triangles (not shown in Figure 4.10) to block all light coming from the light source that does not pass through the holes in G and B or the corresponding holes in the orthogonal copy of the construction. Concluding, we have the following theorem.

Theorem 4.5.3. *Let T be a 3D scene with n triangles, and let t be a triangle of T. Then* $\mathrm{WVM}(t, T)$ *has complexity* $\Theta(n^7)$ *in the worst case.*

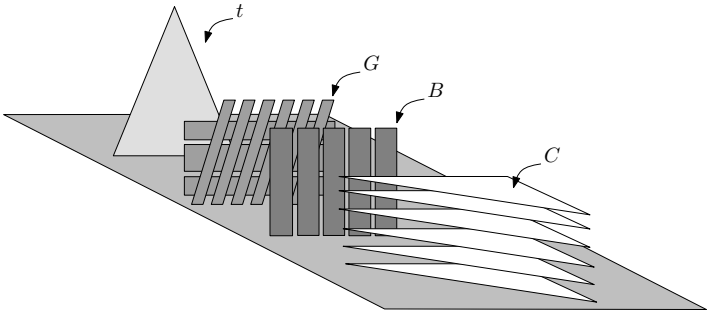


Figure 4.10: The total lower bound construction.

4.5.3 Algorithm

We present an $O(n^7 \log n)$ time algorithm to compute $\mathrm{WVM}(t, T)$. Take one triangle Δ from T and select and all triangles of T that intersect the convex hull of t and Δ. This gives a set T' of triangles that lie (entirely or partially) in between t and Δ. Next, we compute the arrangement $\mathcal{A}$ of all reguli $\mathcal{R}(e, f, g)$ on Δ, where e, f, g are edges from triangles in $T' \cup t$. This two-dimensional arrangement $\mathcal{A}$ is induced by $O(n^3)$ curves on Δ. As in Section 4.4.3, we can compute $\mathcal{A}$ in an output-sensitive way, i.e., in $O((n^3 + K) \log n)$ time, where K is the complexity of $\mathcal{A}$, which is $O(n^6)$ in the worst case. However, again this will not give us an output-sensitive algorithm to compute the visibility map, since not all reguli that occur in $\mathcal{A}$ actually contribute to $\mathrm{WVM}(t, T)$.

Choose a point p in some cell c in $\mathcal{A}$, and compute the arrangement $\mathcal{E}$ of edges of T' projected perspectively from p onto the plane supporting t. This is also the arrangement of projected triangles of T' if they were transparent. The two-dimensional arrangement $\mathcal{E}$ is induced by $O(n)$ line segments and can be computed in $O(n^2)$ time [98]. Because $\mathcal{A}$ corresponds directly to the aspect graph on Δ, the combinatorial structure of $\mathcal{E}$ is the same for any point in c. It only changes when we move p in $\mathcal{A}$ such that p crosses a regulus $\mathcal{R}(e, f, g)$ and enters a cell c' that is adjacent to c. In $\mathcal{E}$, the following happens. The projected edges of e, f, g must have had subedges that formed a triangle in $\mathcal{E}$, which collapses when p reaches $\mathcal{R}(e, f, g)$. When p enters c', an inverted triangle appears between subedges of the projections of e, f, g. Alternatively, two pairs of faces of $\mathcal{E}$ can split or merge when p crosses a regulus that is defined by two edges of the same triangle.

After computing $\mathcal{E}$ for point p in c, we compute some more information. For every cell in $\mathcal{E}$, we compute how many triangles cover that cell, and store this information with the cell. We also maintain whether the cell lies inside or outside t; recall that $\mathcal{E}$ lies in the plane supporting t. For every projected edge e, we maintain a balanced binary search tree $\mathcal{T}_e$ on its intersections with other projected edges. Leaves storing these intersections also contain a pointer into the structure for $\mathcal{E}$; it provides efficient access. Finally, we maintain a global count I of the number of cells of $\mathcal{E}$ that lie inside t and

are covered by zero projected triangles, i.e., the number of cells of $\mathcal{E}$ that illuminate the cell c. If for a cell c of $\mathcal{A}$ the global count is positive, then it is illuminated; if the global count is zero it is dark. See Figure 4.11 for an illustration of the various structures that the algorithm incorporates.

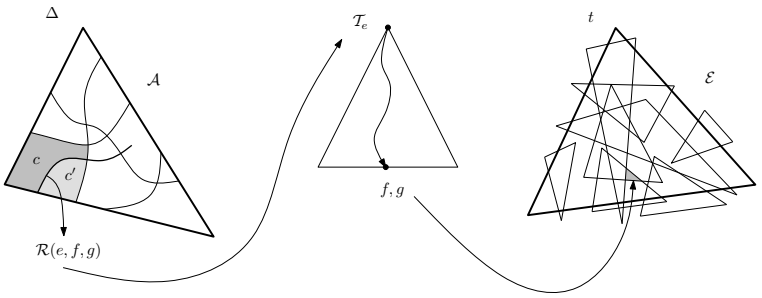


Figure 4.11: Illustration of the algorithm that computes the part of $\mathrm{WVM}(t, T)$ on a triangle Δ.

All of the above information can be updated when we go from cell c to a neighboring cell c' in $O(\log n)$ time. We must exchange a pair of leaves in three binary search trees, we have to make local changes to $\mathcal{E}$ for the collapsing and/or appearing faces, we must determine the number of projected triangles that cover the new cells in $\mathcal{E}$, and we have to update the global count I of 'illuminating' cells of $\mathcal{E}$. The number of projected triangles that cover the new faces in $\mathcal{E}$ has the corresponding value of the disappearing face, plus or minus 1 or 3. This can be determined from the edges e, f, g of T'. The value of I can decrease or increase by 1, or stay the same. Only the changes of the binary trees takes $O(\log n)$ time, the other updates require only $O(1)$ time, after which we can label c' either illuminated or dark. If $\mathcal{A}$ has complexity K, which is $O(n^6)$ in the worst case, the total update time if we traverse all cells of $\mathcal{A}$ is $O(K \log n)$, after which we have obtained the weak visibility map of t on Δ. Repeating this procedure for all n triangles of T gives us $\mathrm{WVM}(t, T)$.

Theorem 4.5.4. *Let T be a 3D scene with n triangles, and let t be a triangle of T. The weak visibility map $\mathrm{WVM}(t, T)$ can be computed in $O(n^7 \log n)$ time.*

4.6 Concluding remarks

We presented lower and upper bounds on the combinatorial complexities of various types of visibility maps of edges and triangles in a 3D scene. As can be seen in Table 4.1, there are a few gaps left between lower and upper bounds. Finally, considering the discussion in Sections 4.4.3 and 4.5.3 about the output-sensitive computation of arrangements of reguli, it would be interesting to find algorithms that are indeed output-sensitive with respect to the complexity of the visibility map in question.

Chapter 5

On realistic terrains

The contents of this chapter will be published as:

E. Moet, M. van Kreveld, and A. F. van der Stappen.
On realistic terrains.
Computational Geometry: Theory and Applications,
Y. Emiris and L. Palios (Eds.), Special Issue: Selected Papers from the 22nd Annual European Workshop on Computational Geometry (EWCG'06), 2008 or 2009.

Abstract

We study worst-case complexities of visibility and distance structures on terrains under realistic assumptions on edge length ratios and the angles of the triangles, and a more general low-density assumption. We show that the visibility map of a point for a realistic terrain with n triangles has complexity $\Theta(n\sqrt{n})$. We also prove that the shortest path between two points p and q on a realistic terrain passes through $\Theta(\sqrt{n})$ triangles, and that the bisector of p and q has complexity $O(n\sqrt{n})$. We use these results to show that the shortest path map for any point on a realistic terrain has complexity $\Theta(n\sqrt{n})$, and that the Voronoi diagram for any set of m points on a realistic terrain has complexity $\Omega(n + m\sqrt{n})$ and $O((n + m)\sqrt{n})$. Our results immediately imply more efficient algorithms for computing the various structures on realistic terrains.

5.1 Introduction

One of the main objectives of computational geometry is to uncover the computational complexity of geometric problems. It provides a theory that explains how efficiently geometric problems that arise in applications can be solved. However, in many cases a discrepancy exists between the provable worst-case computational complexity of an algorithm and the actual running time behavior of that algorithm on inputs that arise in applications. This has led to the study of *fatness* and *realistic input models*. By making assumptions on the input, certain hypothetical worst-case scenarios can no longer occur, and a more efficient solution to a problem can be shown for all inputs that satisfy the assumptions. Either an existing algorithm can be shown to be more efficient for realistic inputs, or a special algorithm is designed that assumes that it will run on realistic inputs only. Our algorithmic results are of the first type.

Among the first applications of realistic input models in computational geometry, Alt et al. [1] considered motion planning for a rectangular robot. The efficiency depends on the aspect ratio of this rectangle. Matoušek et al. [130] showed that if all triangles of a set of n triangles have their angles bounded away from zero (at least α, for some constant $\alpha > 0$), then the union of these triangles has $O(n)$ holes rather than $O(n^2)$ for the general case, and the boundary complexity of the union is $O(n \log\log n)$. Such triangles are called *fat*. Since then, various definitions of fatness [6, 62, 153, 114] have been proposed. Other realistic models—such as low density [182], unclutteredness [192], and simple-cover complexity [133]—consider the spatial distribution of objects or their features. An overview of reduced combinatorial complexities and improved algorithmic efficiencies for inputs satisfying these models was given in [22, 192] along with a model hierarchy. Most papers concern union complexities of fat objects (e.g. [6, 62]) or motion planning in realistic environments (e.g. [21, 182, 181]), but many other results exist as well. For example, de Berg studied linear-size binary space partitions for uncluttered scenes [14] and vertical ray shooting in fat objects [16], Mitchell et al. [133] considered ray shooting in scenes with small simple cover complexity, Overmars and van der Stappen [153] and Vleugels and Schwarzkopf [172] considered point location and range searching in sets of fat objects and low density scenes respectively, and Erickson [67] studied Boolean

combinations and Minkowski sums of polygons and polyhedra whose features satisfy certain realistic locality assumptions.

Realistic assumptions have not yet been studied for polyhedral terrains. However, several geometric structures on terrains have complexities much higher than typical in applications. For example, the visibility map of a point p for a polyhedral terrain of n triangles, i.e., the subdivision of the terrain into maximal connected components such that throughout each cell the viewpoint p is either visible or invisible, has worst-case complexity $\Theta(n^2)$ [131]. The shortest path between two points on a terrain passes through $\Theta(n)$ triangles in the worst case, and even the bisector of two points on a terrain can have quadratic size. Hence, the discrepancy between theoretical complexity bounds and typical complexity bounds exists on terrains as well. In this chapter, we analyze this discrepancy by studying realistic assumptions on terrains.

For visibility maps, we give three assumptions whose combination provides an $\Theta(n\sqrt{n})$ bound on the worst-case complexity visibility map of a terrain for both views from infinity and perspective views. In fact, our results apply to the transparent visibility map, that is, our bounds hold even when occlusion is not taken into account. Dropping any of the three assumptions immediately makes an $\Omega(n^2)$ lower bound construction possible. It is interesting to note that the assumptions all refer to the xy-projection of the terrain.

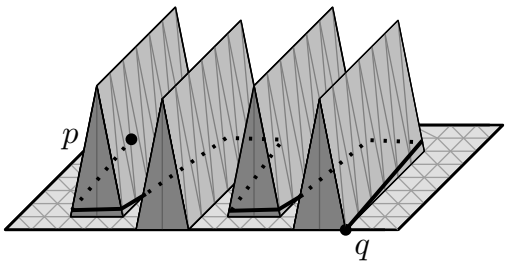


Figure 5.1: The shortest path between p and q crosses $\Omega(n)$ triangles.

Next, we consider distance structures on terrains. Using only the three assumptions for visibility, we can still have shortest paths that have linear complexity; the terrain in Figure 5.1 can be constructed such that in the projection, all its vertices lie on a regular grid. Therefore, we introduce a fourth assumption that relates to the steepness of the terrain, and show that any shortest path between two points passes through only $\Theta(\sqrt{n})$ triangles. For a bisector of two points, we show that the same set of four assumptions gives an $O(n\sqrt{n})$ complexity bound rather than quadratic. We give an $\Omega(n)$ size lower bound. The shortest path map for a source point s is the subdivision of the terrain into regions where the combinatorial structure of shortest paths from s is the same. In general, it has worst-case complexity $\Theta(n^2)$, but we show that under our assumptions it is $\Theta(n\sqrt{n})$.

Finally, we study Voronoi diagrams for m point sites on terrains. The Voronoi diagram for a set of sites is the subdivision of the terrain into cells such that all points

in one cell have the same closest site, where the distance between two points on the terrain is the length of the shortest path between these two points. In general, if m is $O(n)$, the Voronoi diagram also has quadratic worst-case complexity; this is even the case for only two sites. Our assumptions allow us to prove an upper bound of $O((n+m)\sqrt{n})$, and we give a lower bound of $\Omega(n+m\sqrt{n})$.

The better bounds on the combinatorial complexity of the various structures in most cases also imply improved computation times. The output-sensitive construction of the visibility map of a terrain by Katz et al. [110] implies that for realistic terrains, it can be computed in $O(n\sqrt{n}\log n)$ time. For the shortest path maps and Voronoi diagrams, the algorithm of Mitchell et al. [132] leads to $O(n\sqrt{n}\log n)$ time bounds for their construction on realistic terrains; see Sections 5.4.2 and 5.4.3 for more details.

This chapter is structured as follows. In Section 5.2, we list three assumptions on realistic terrains and argue that no subset of these three is sufficient to prove a subquadratic upper bound on the complexity of the visibility map. Moreover, we argue why we believe our assumptions are reasonable and present an indication of the values of the model parameters in terrains that arise in practice. We also prove several auxiliary results using these assumptions. In Section 5.3, we obtain a $\Theta(n\sqrt{n})$ bound on the size of the visibility map for both views from infinity and perspective views. In Section 5.4, we add a fourth assumption to study shortest paths, bisectors, and the shortest path map; we give a $\Theta(n\sqrt{n})$ size bound for the latter structure in Section 5.4.2. In Section 5.4.3, we show that the Voronoi diagram of m sites has worst-case complexity $\Omega(n+m\sqrt{n})$ and $O((n+m)\sqrt{n})$ on realistic terrains. In Section 5.5, we give the conclusions and open problems.

5.2 Input model

Let $\mathcal{T}$ be a polyhedral terrain: a piecewise-linear continuous function defined over the triangles of a triangulation in the xy-plane. A terrain $\mathcal{T}$ comprises a set T of n triangles, a set E of n_e edges, and a set V of n_v vertices, where n_e and n_v are $O(n)$. Throughout this section, with the exception of Section 5.2.3, we refer to the orthogonal projection of $\mathcal{T}$ onto the xy-plane. We give two sets of assumptions on the properties of the projection of $\mathcal{T}$ that allow us to prove a subquadratic upper bound on the complexity of the visibility map. Below, we give the set of assumptions that we used in an earlier version of this chapterr [138]; we prove the results from [138] in this chapter with a weaker set of assumptions, which we introduce further on in this section.

1. the minimum angle of any triangle in T in the projection to the xy-plane is at least α,
2. the smallest rectangle that contains the projection of the terrain has side lengths 1 and c, and
3. the longest xy-projection over all edges in E is at most d times as long as the shortest one.

The values α, c, and d in the above assumptions are positive constants. Note that the rectangle in Assumption 2 is not necessarily axis-aligned. A projected triangle

that satisfies the first assumption is *fat* [130]. We used fatness of the triangles in the projection; alternatively, we could assume that the angles of the triangles in 3D are all larger than some $\alpha > 0$. However, as will become clear from Figure 5.5 in the following section, this assumption does not help to obtain a subquadratic complexity bound for the visibility map.

We could have stated the second assumption as an aspect ratio condition as well, i.e., the ratio of the side lenghts of the smallest rectangle that contains the projection of $\mathcal{T}$ is $1 : c$. All results in this chapter then hold with a multiplicative factor. However, since the first and third assumption are scale-independent, we can always scale down the terrain so that the shortest side of the containing rectangle has unit length. Therefore, we use this simplified assumption in the rest of the chapter.

Assumption 3 seems quite strong, but it turns out to be crucial to obtain a non-trivial lower bound on the minimum edge length, an auxiliary result that we use extensively in this chapter. All angles in the triangulation in Figure 5.2 are α-fat for $\alpha = \frac{\pi}{4}$, its boundary side lengths are 1 and 2, and it consists of at most $3n$ triangles. The edge length ratio is not bounded by a constant, which makes it possible that edges of the triangulation have length $O(\frac{1}{n})$ or $\Omega(1)$.

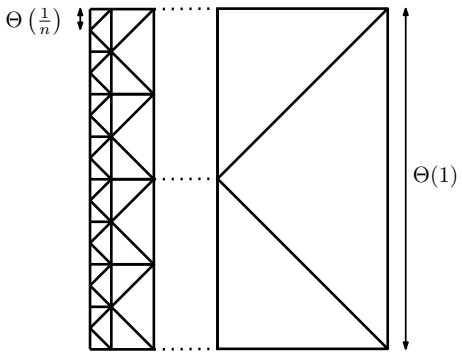


Figure 5.2: A low-density triangulation does not necessarily induce a bounded edge length ratio.

Since fatness probably is an overly restrictive assumption—it is unlikely that a constant number of non-fat triangles will induce a quadratic-size visibility map—we now consider a weaker assumption on the input, which we use in the remainder of this chapter to prove subquadratic upper bounds on the combinatorial complexity of various structures. Moreover, we show that any fat triangulation satisfies this weaker assumption; therefore, all our results hold for triangulations with fat triangles as well.

Because our input is a planar triangulation instead of a general set of objects, we have adapted the original definition of low-density in [192] to the following tailored version.

Definition 5.2.1. *Let k be a positive integer, let T be a planar triangulation with n triangles, and let E be the set of $O(n)$ edges of T. We call T a* k-low-density *triangula-*

tion if for any axis-aligned square S with side length s, the number of edges in E with length greater than or equal to s that intersect S is at most k. The smallest k for which T is a k-low-density triangulation is the density *of T.*

Lemma 5.2.1. *Let T be a triangulation in the plane in which each angle of each triangle is at least $\alpha > 0$. Then T is a k-low-density triangulation for some $k = O(\frac{1}{\alpha})$.*

Proof: We have to show an upper bound on the number of edges of T of length at least ℓ that intersect an arbitrary axis-aligned square B_ℓ with side length ℓ. Each edge that intersects B_ℓ belongs to some triangle t of T, where all three angles in t are at least α. Let B^* be the box that is B_ℓ scaled by a factor 3 with respect to its center; see Figure 5.3.

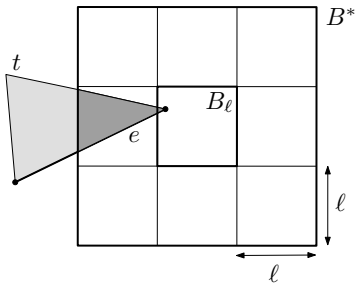


Figure 5.3: Illustration for the proof of Lemma 5.2.1.

Let t be a triangle that has an edge e of length at least ℓ which intersects B_ℓ. By the fatness of t, the area of $B^* \cap t$ is at least $\frac{1}{2}\ell^2 \sin\alpha$. This implies that there are at most

$$\frac{18\ell^2}{\ell^2 \sin\alpha}$$

such triangles, and thus the density of T is at most

$$\frac{54}{\sin\alpha} = O\left(\frac{1}{\alpha}\right).$$

This result holds for all $0 < \alpha < \frac{\pi}{2}$. ∎

Note that the triangulation in Figure 5.2 is a 7-low-density triangulation. From now on, we assume that $\mathcal{T}$ satisfies the following three properties:

1a. the projection of $\mathcal{T}$ onto the xy-plane is a k-low-density triangulation,
2. the smallest rectangle that contains the projection of the terrain has side lengths 1 and c, and
3. the longest xy-projection over all edges in E is at most d times as long as the shortest one.

The values k, c, and d in the above assumptions are positive constants.

Definition 5.2.2. *We call a polyhedral terrain $\mathcal{T}$ a* realistic terrain *if $\mathcal{T}$ satisfies Assumptions 1a, 2, and 3.*

Although we are aware that the use of the term *realistic* might spark some discussion, we do employ the term realistic terrain to associate our assumptions with the widely accepted term *realistic input model*. Below, we argue why we believe that our assumptions are reasonable, and give an indication of the values of our model parameters in practice.

Low-density of input scenes for geometric algorithms has been studied extensively. De Berg et al. [22] conducted experiments in which they computed the density parameter of terrains that arise in practice. They give results for TINs representing certain areas in the US, which they constructed by extracting the most important points from DEM-data by using the VIP selection method [39], and consecutively generating the Delaunay triangulation of these points. The number of vertices of the scenes they consider ranges from 100 to 5000. Their results indicate that the density of these scenes varies between approximately fifteen and twenty-five, and is independent of the number of triangles of the terrain. Although the definition of density in [22] is slightly different from the one that we use in this chapter, the actual density parameter can be shown to differ by only a small constant factor.

The second assumption, constant aspect ratio of the smallest rectangle that contains the projection of the terrain, guarantees that the terrain cannot be unnaturally long and/or skinny, which is quite reasonable to assume, especially since most data sets from GIS are measurements over (almost) square regions of land, and thus fulfill this condition by construction.

For the third assumption, we have recently conducted experiments on various TINs which were constructed from USGS data [191]. From DEM data sets, we created a terrain with a given number of vertices, by repeatedly adding points to a Delaunay triangulation in the xy-plane, where we always added the point with the largest deviation from the current (interpolated) approximation [71, 83, 102]. We investigated nine such terrains, with values of n varying from 100 to 7000. Recall that Assumption 3 states that there exists a constant upper bound d on the edge length ratio. When terrains are constructed from DEM data, where height values are measured at regular intervals, there is a natural upper bound on the edge length ratio. This natural upper bound depends on the smallest xy-distance between two points from the DEM data and on the value of c (Assumption 2). In our experiments, we found that for all nine TINs, the value d never increased for n larger than 700. For n between 1000 and 2000, the edge length ratio showed a rapid decrease, and for seven of the nine TINs, for n greater than 3000, the edge length ratio was always below 100.

5.2.1 No subset of assumptions is sufficient

In this section, we show that if we drop any of the three assumptions above, it is possible to construct a quadratic-size visibility map. Thus, no subset of the assumptions is sufficient to give a subquadratic upper bound.

A vertex of the visibility map of a point p directly corresponds to a line through p that is a common tangent of two terrain edges. The terrain that is displayed in Figure 5.4 produces an $\Omega(n^2)$ size visibility map; the $\Omega(n)$ hills in the front induce a linear size subdivision on each of the $\Omega(n)$ triangles in the back.

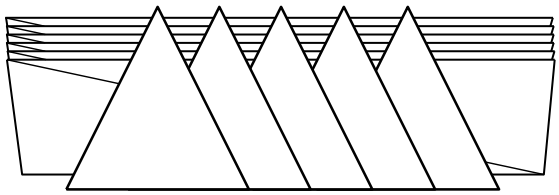


Figure 5.4: An $\Omega(n^2)$ complexity visibility map of a terrain for a viewpoint at infinity.

First, if we allow the projected triangulation to have non-constant density, we can place two sets of $\Omega(n)$ triangles in a rectangular region of constant area in such a way that *every* triangle in the first set produces a constant number of visibility map vertices with *every* triangle of the second set, which results in a visibility map of complexity $\Omega(n^2)$. A schematic illustration of such a construction is shown in Figure 5.5; the Θ's refer to distances in the xy-projection. Note that the triangles in this construction are not α-fat for any constant $\alpha > 0$, and that the same construction can be achieved with a terrain that satisfies Assumptions 2 and 3, and in which the triangles are fat in 3D, but not in the projection.

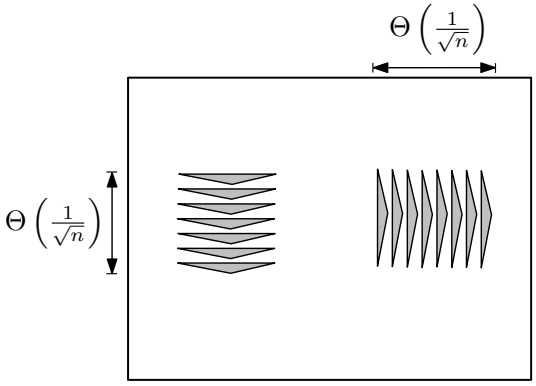


Figure 5.5: A terrain that produces a quadratic-size visibility map if the viewpoint is located infinitely far to the left. The terrain satisfies Assumptions 2 and 3.

To obtain a valid terrain from this construction, we need to triangulate the white parts in Figure 5.5 such that the resulting triangles do not violate Assumption 3. The subconstruction that achieves this is shown in Figure 5.6; we need a rectangular region

with side lengths $O\left(\frac{\log n}{\sqrt{n}}\right)$ and $O\left(\frac{1}{\sqrt{n}}\right)$ to adjoin the two subconstructions shown in Figure 5.5. This construction uses $O(n)$ triangles. Similar constructions in which the width of the triangles grows, can be used to triangulate the other white parts of Figure 5.5 and thus connect the two subconstructions with the boundary.

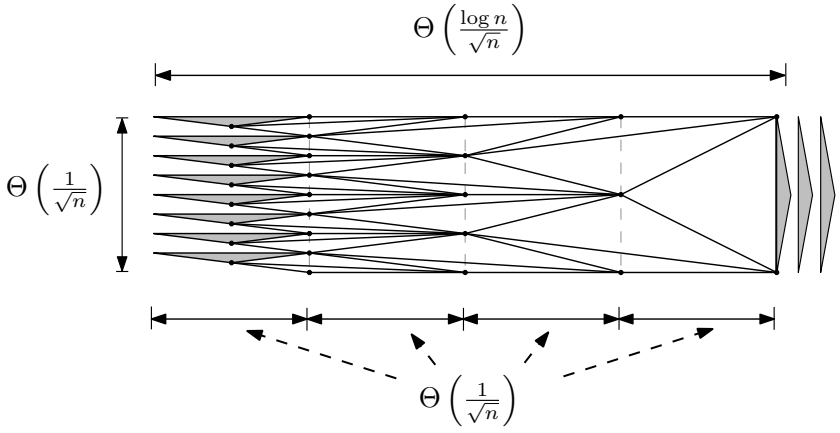


Figure 5.6: The subconstruction used to obtain a valid terrain from Figure 5.5.

We can also construct a visibility map of complexity $\Omega(n^2)$ for a terrain that obeys the first and third assumption, but not the second one. Since the aspect of the containing rectangle ratio is no longer a constant, we can again place two sets of $\Omega(n)$ triangles that mutually interact in the visibility map; see Figure 5.7. Because the area that we are able to use is not bounded by a constant, it is straightforward to triangulate the rest of the terrain with $O(n)$ triangles, while not violating Assumptions 1a and 3.

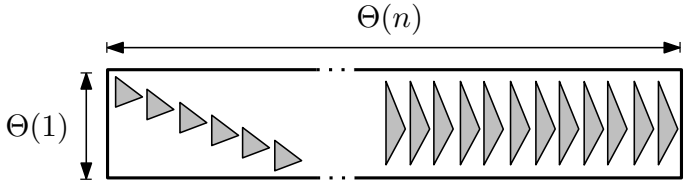


Figure 5.7: A terrain that produces a quadratic-size visibility map if the viewpoint is located infinitely far to the left. The terrain satisfies Assumptions 1a and 3.

Finally, if we allow an unbounded edge length ratio, it is again possible to construct a terrain with a visibility map of complexity $\Omega(n^2)$. A schematic display of this construction is shown in Figure 5.8.

Again, we need to triangulate the white parts of Figure 5.8 to complete the terrain, but now we can only use $O(n)$ triangles, while not violating Assumption 1a. In order

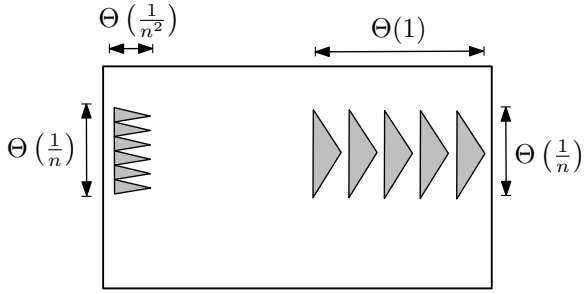


Figure 5.8: A terrain that produces a quadratic-size visibility map if the viewpoint is located infinitely far to the left. The terrain satisfies Assumptions 1a and 2.

to adjoin the two sets of triangles shown in Figure 5.8, we need to grow the lengths of the shortest edges by a factor n. We use a distance of $O\left(\frac{1}{n}\right)$ and $O(n)$ triangles to achieve this. This subconstruction is shown in Figure 5.9. The other white parts in Figure 5.8 can also be triangulated with $O(n)$ triangles, such that we do not violate Assumption 1a, by letting the edge lengths grow in a similar way.

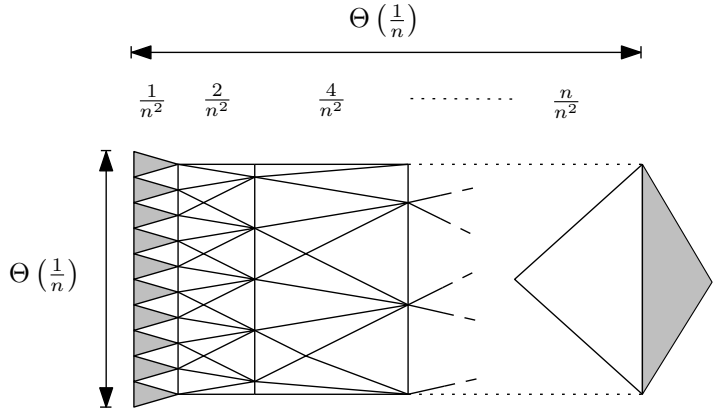


Figure 5.9: The subconstruction used to obtain a valid terrain from Figure 5.8.

Finally, we observe that in Figure 5.8, the ratio of edge lengths is $O(n)$, and hence the assumption of Erickson [67] of polynomially bounded edge lengths is not strong enough in our case.

5.2.2 Auxiliary results

We now give several auxiliary lemmas on properties of the *projection* of $\mathcal{T}$, which we use to obtain our main results in Sections 5.3 and 5.4. Throughout this section, R is the

smallest rectangle that contains the projection of $\mathcal{T}$, and x is the length of the shortest edge of the projection of $\mathcal{T}$. We let B_ℓ denote an axis-aligned square (box) with side lengths ℓ, $B_\ell(p)$ denotes the square B_ℓ translated such that the point p is its center, and $\mathcal{E}(B_\ell)$ denotes the set of edges of length at least ℓ that intersect the square B_ℓ.

Lemma 5.2.2. *Each vertex v of V has degree at most k.*

Proof: By Assumption 1a, $\mathcal{E}(B_x(v))$ is a set of at most k edges from E. The lemma follows from the fact that this set includes all edges that are incident to v. ∎

Lemma 5.2.3. *All edges in E have length $\Theta\left(\frac{1}{\sqrt{n}}\right)$.*

Proof: First, we observe that all edges in E have length at most dx. Any triangle in T is at most as large as the equilateral triangle with edge lengths dx. The maximum area of any triangle is therefore

$$\frac{\sqrt{3}(dx)^2}{4}.$$

The sum of the areas of the triangles in T is at most

$$c \leq \frac{n\sqrt{3}(dx)^2}{4},$$

and therefore

$$x \geq \frac{2\sqrt{c}}{\sqrt[4]{3}\ d\sqrt{n}} = \Omega\left(\frac{1}{\sqrt{n}}\right).$$

For the upper bound, it is sufficient to show that x is $O\left(\frac{1}{\sqrt{n}}\right)$, since the longest edge has length dx. To this purpose, we cover R by a set of copies of the square B_x. By Assumption 2, we can do this with less than $\frac{c^2+1}{x^2}$ squares. Because of the low-density assumption, every square B_x intersects at most k edges of E. Because x is the minimum edge length, every single edge that intersects a given B_x is indeed included in $\mathcal{E}(B_x)$. Since every edge intersects a constant number of squares, and since there are $\Omega(n)$ edges in E, we have that $k\frac{\sqrt{1+c^2}}{x^2}$ is $\Omega(n)$, which implies that x is $O\left(\frac{1}{\sqrt{n}}\right)$. ∎

Lemma 5.2.4. *Let $\mathcal{R}$ be a rectangle that intersects the projection of $\mathcal{T}$, of which both side lengths are $\Omega\left(\frac{1}{\sqrt{n}}\right)$, and that has total area A. Then $\mathcal{R}$ intersects $O(knA)$ triangles of T.*

Proof: We cover $\mathcal{R}$ by a set of copies of the square B_x; see Figure 5.10. Since x is $\Omega\left(\frac{1}{\sqrt{n}}\right)$, this is possible with $O(An)$ such squares. Because x is the minimum edge length, every single edge that intersects a given square B_x is indeed included in $\mathcal{E}(B_x)$. By Assumption 1a, we have that $|\mathcal{E}(B_x)| \leq k$ for each B_x. Therefore, the sum of all values $|\mathcal{E}(B_x)|$, and thus the number of edges (and triangles) that intersect $\mathcal{R}$, is $O(knA)$. ∎

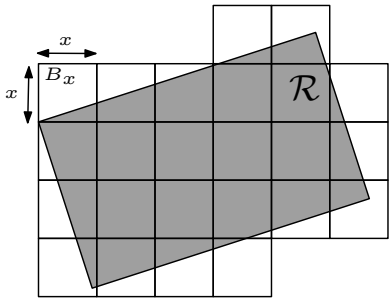


Figure 5.10: Illustration of the proof of Lemma 5.2.4; the rectangle $\mathcal{R}$ is covered by copies of B_x.

Lemma 5.2.5. *Let s be a straight line segment that intersects the projection of $\mathcal{T}$. Then s intersects $O(k\sqrt{n})$ triangles of T.*

Proof: By Assumption 2, the length of s within R is $O(1)$. Let L be a slab of width $\Theta\left(\frac{1}{\sqrt{n}}\right)$, enclosed by two lines parallel to s, and which contains s strictly in its interior. All edges that intersect s must intersect L as well. By Assumption 2, the area of L is $O\left(\frac{1}{\sqrt{n}}\right)$, and so, by the previous lemma, L intersects $O(k\sqrt{n})$ triangles. ∎

5.2.3 Dihedral angles

So far, we have only considered assumptions that refer to the projected triangulation induced by a terrain. All complexity bounds for the visibility map of a realistic terrain in Section 5.3 are achieved with only the first three assumptions, which only deal with properties of the terrain in the projection. In particular, we do not need a bound on the dihedral angles to prove the upper bounds, and the lower bound constructions are all possible with bounded dihedral angle.

The terrain in Figure 5.1 does satisfy Assumptions 1a, 2, and 3, but a shortest path between two points on the terrain still passes through $\Theta(n)$ triangles in the worst case. In Section 5.4, we discuss distance structures on terrains, and to bound their complexity, we introduce an additional assumption:

4. The dihedral angle of the supporting plane of any triangle in T with the xy-plane is at most β, where $\beta < \frac{\pi}{2}$ is some constant.

Assumption 4 implies that the maximum slope of a line segment on any triangle of $\mathcal{T}$ is $\tan\beta = O(1)$. In this section and from Section 5.4 onwards, we call a terrain *a realistic terrain* if it satisfies all four assumptions. It is easily verified that no subset of our four assumptions is sufficient to obtain a sublinear bound on the complexity of a shortest path; for each subset of three assumptions, we can create an $\Omega(n)$ lower bound.

In our experiments, described in the paragraph before Section 5.2.1, we also computed the maximum dihedral angle. As before, we considered TINs for nine DEM data sets, with values of n in the range from 100 to 7000. For all terrains the value of β appeared to be independent of n. Moreover, for six out of the nine data sets, the constructed TINs that had 4000 or more triangles, consisted of triangles that all had a supporting plane with dihedral angle below 70°.

Lemma 5.2.6. *The shortest path $P(p,q)$ between two points p and q on a realistic terrain $\mathcal{T}$ has $O(1)$ length.*

Proof: Let $P'(p,q)$ be the path from p to q over $\mathcal{T}$ whose projection is the straight line segment $\overline{pq}$. Obviously, $P(p,q)$ is at most as long as $P'(p,q)$. By Assumption 4, the length of $P'(p,q)$ is at most $\tan\beta \cdot |\overline{pq}|$. By the second assumption, $\overline{pq}$ has length at most $\sqrt{1+c^2}$, and thus the length of $P(p,q)$ is bounded from above by $\tan\beta \cdot \sqrt{1+c^2}$, which is $O(1)$. ∎

Lemma 5.2.7. *The shortest path $P(p,q)$ between two points p and q on a realistic terrain $\mathcal{T}$ crosses $O(k\sqrt{n})$ triangles.*

Proof: First we observe that $P(p,q)$ intersects any triangle at most once, and within each triangle, $P(p,q)$ is a straight line segment. We cover the projection of $P(p,q)$ completely by copies of the square B_x. The first square that we place is $B_x(p)$. The next square is $B_x(p')$, where p' is the point on the projected path that is closest to p and which has not yet been covered. We keep placing squares in this way until the entire projection of $P(p,q)$ is covered. See Figure 5.11 for an example of such a covered path.

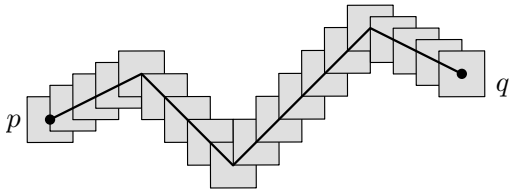


Figure 5.11: Covering the path $P(p,q)$ by squares with side length x.

Every square covers at least the length $\frac{1}{2}x$ of the projection of $P(p,q)$. By the previous lemma, and because x is $\Omega\left(\frac{1}{\sqrt{n}}\right)$, we can do this with $O(\sqrt{n})$ such squares. By Assumption 1a, each square B_x intersects at most k edges of E, and thus $P(p,q)$ crosses $O(k\sqrt{n})$ triangles. ∎

It is easy to see that on any realistic terrain $\mathcal{T}$, two points exist whose shortest path passes through $\Omega(\sqrt{n})$ triangles, and thus we conclude this section with the following lemma.

Lemma 5.2.8. *Let $\mathcal{T}$ be a realistic terrain with n triangles. The shortest path over $\mathcal{T}$ between two points p and q on $\mathcal{T}$ passes through $\Theta(\sqrt{n})$ triangles in the worst case.*

5.3 The visibility map

5.3.1 Upper bounds

Let p be the viewpoint of the visibility map $\mathrm{VM}(p, \mathcal{T})$ that we consider. Throughout this section, we study the transparent visibility map, that is, we do not take occlusion by the terrain triangles into account. For a given edge e, let I_e be the set of terrain edges with which e can *interact*, i.e., with which it can create vertices of $\mathrm{VM}(p, \mathcal{T})$, and which lie closer to the viewpoint than e. For every edge f with which e creates a vertex of $\mathrm{VM}(p, \mathcal{T})$, there exists a line-of-sight ℓ that passes through p and first is tangent to f and then to e; see Figure 5.12. Let t be a triangle of $\mathcal{T}$ that ℓ intersects. The point of intersection is a vertex of $\mathrm{VM}(p, \mathcal{T})$; it is the intersection of the perspective projections e' (of e) and f' (of f), with respect to p, on the triangle t.

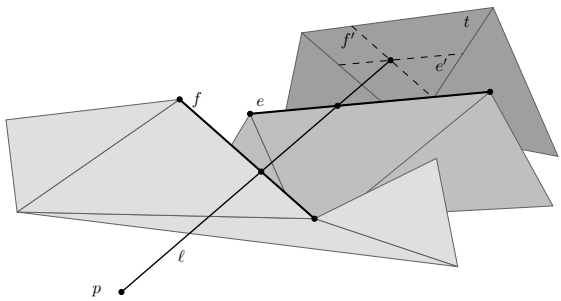


Figure 5.12: The line-of-sight ℓ is tangent to the edges f and e and creates a vertex of $\mathrm{VM}(p, \mathcal{T})$ on the triangle t.

We bound the combinatorial complexity of $\mathrm{VM}(p, \mathcal{T})$ by giving an upper bound on the cardinality of I_e. Since any two edges create $\Theta(1)$ vertices of $\mathrm{VM}(p, \mathcal{T})$ in the worst case, we get the following expression.

$$\text{Complexity of } \mathrm{VM}(p, \mathcal{T}) = O\left(\sum_{e \in E} \#I_e\right) \tag{5.1}$$

For an edge e, we define the *influence region* of e as the locus of the edges that (i) lie in between p and e, and (ii) with which e interacts in the (transparent) visibility map of p. This definition implies that we charge a visibility map vertex to the *last* edge that the corresponding tangent touches. We denote the influence region of an edge e by $\mathcal{R}_e$, and we give an upper bound on its area. Using this bound, we can bound $\#I_e$ from above by the number of edges in E whose projection intersects $\mathcal{R}_e$. We distinguish two

cases based on the location of the viewpoint: (i) p is located infinitely far away from $\mathcal{T}$ (parallel projection), and (ii) p does not lie at infinity (perspective projection).

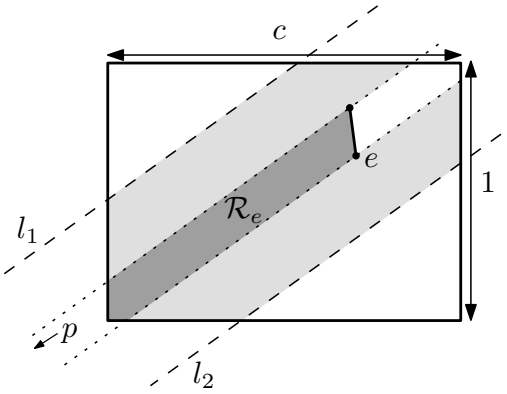


Figure 5.13: The influence region for an edge e and a viewpoint p at infinity.

If the viewpoint p is located at infinity, then for any edge e, in the projection to the xy-plane, all triangles that intersect the influence region $\mathcal{R}_e$ lie either entirely within the influence region, or partly in the influence region and partly outside this region, but in a buffer region that is enclosed between two parallel lines. Figure 5.13 illustrates this; please note that the proportions in this figure are distorted for illustration purposes. Let l_1 and l_2 be the two lines that pass through the two endpoints of e and are parallel to the viewing direction induced by the viewpoint p. Now we translate l_1 and l_2 in the perpendicular direction, away from each other, over a distance of the length of the longest edge. Let L_e be the slab that is bounded by l_1 and l_2; it is easy to verify that L_e completely contains all the triangles that intersect $\mathcal{R}_e$. By Lemma 5.2.3, for any edge e from E, $L_e \cap R$ is completely contained in a rectangle of area $O\left(\frac{1}{\sqrt{n}}\right)$, and thus, by Lemma 5.2.4, the number of triangles (and edges) in L_e is $O(\sqrt{n})$. Using Equation (5.1), we obtain the following lemma.

Lemma 5.3.1. *Let $\mathcal{T}$ be a realistic terrain with n triangles, and let p be a viewpoint at infinity. Then $\mathrm{VM}(p, \mathcal{T})$ has complexity $O(n\sqrt{n})$ in the worst case.*

The influence region for an edge e and a perspective view is very similar to the influence region of e and some viewpoint at infinity; see Figure 5.14. In fact, for some viewing direction, $\mathcal{R}_e$ is contained in the influence region from infinity, and hence $\mathcal{R}_e$ has area $O\left(\frac{1}{\sqrt{n}}\right)$. The two lightly shaded buffer regions around $\mathcal{R}_e$ are both enclosed by a pair of parallel lines at a distance of the longest (projected) edge of E of each other. By Lemma 5.2.3, this distance is $\Theta\left(\frac{1}{\sqrt{n}}\right)$, which together with Assumption 2 gives that both buffers have area $O\left(\frac{1}{\sqrt{n}}\right)$ as well. Thus, by Lemma 5.2.4, the number

of triangles (and edges) in that intersect $\mathcal{R}_e$ in this case is also $O(\sqrt{n})$, which, together with Equation (5.1), yields the lemma below.

Lemma 5.3.2. *Let $\mathcal{T}$ be a realistic terrain with n triangles, and let p be a point located on or above $\mathcal{T}$. Then $\mathrm{VM}(p, \mathcal{T})$ has complexity $O(n\sqrt{n})$ in the worst case.*

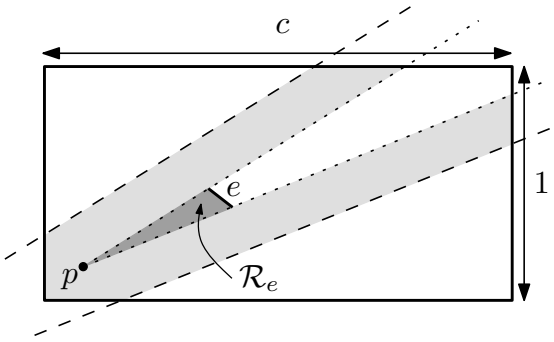


Figure 5.14: The influence region for an edge e and a viewpoint p close to or above the terrain. (Proportions are distorted for illustration purposes.)

Note that in the figure above, we displayed the projection of the viewpoint inside the projection of the terrain, but this is not necessarily the case. If we move the viewpoint further and further away from the terrain, the perspective influence region converges into the parallel influence region.

5.3.2 Lower bounds

In this section, we show that the upper bounds from the previous section are tight.

We start by describing a construction of a visibility map for parallel views that has $\Omega(n\sqrt{n})$ vertices. Under the assumptions of Section 5.2, we can place $\Theta(\sqrt{n})$ triangles in a rectangle of constant length and of width $\Theta(\frac{1}{\sqrt{n}})$. This can be done in such a way that these $\Theta(\sqrt{n})$ triangles together form a smaller version of the construction in Figure 5.4; in the front, we place $\Omega(\sqrt{n})$ triangles, each of which interacts with all $\Omega(\sqrt{n})$ triangles in the back. For such a rectangle, the visibility map for a point at infinity has complexity $\Omega(n)$.

Since the projection of $\mathcal{T}$ is a rectangle with side lengths 1 and c, we can replicate this construction $\Omega(\sqrt{n})$ times; see Figure 5.15 for a schematic illustration. If the viewpoint is located infinitely far down, this terrain has a visibility map of complexity $\Omega(n\sqrt{n})$.

Lemma 5.3.3. *The visibility map of a realistic terrain for a viewpoint at infinity can have $\Omega(n\sqrt{n})$ vertices.*

For perspective views, we show a very similar lower bound. By a transformation of the construction in Figure 5.4, we can create a subconstruction with $\Omega(\sqrt{n})$ triangles

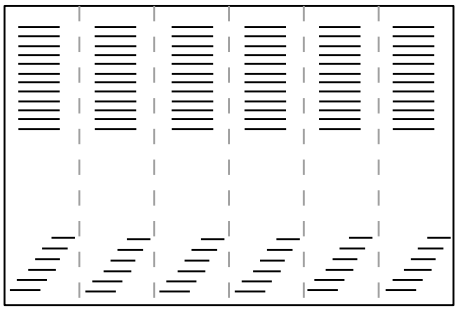


Figure 5.15: The visibility map of a viewpoint at infinity can have $\Omega(n\sqrt{n})$ vertices.

that are located in a truncated wedge of area $\Theta(\frac{1}{\sqrt{n}})$, instead of in a rectangle of the same area, as we did in Figure 5.15. Note that this is possible with such a transformation that the edge lengths and angles of the transformed triangles are all within a constant factor of the original values. If we place the viewpoint at the apex of the (non-truncated) wedge, then this subconstruction produces a visibility map with $\Omega(n)$ vertices. We can replicate this construction $\Omega(\sqrt{n})$ times in a rectangle of $\Theta(1)$ area; Figure 5.16 displays the total construction schematically.

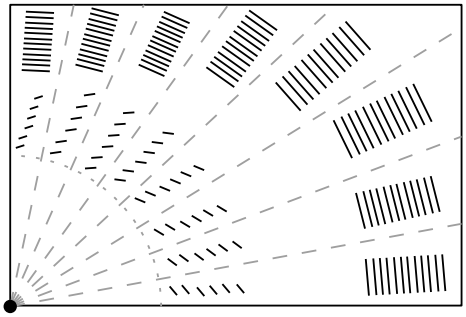


Figure 5.16: The perspective visibility map can have $\Omega(n\sqrt{n})$ vertices.

Lemma 5.3.4. *The visibility map of a realistic terrain for a perspective view can have* $\Omega(n\sqrt{n})$ *vertices.*

We conclude with a theorem that summarizes the results of this section.

Theorem 5.3.1. *Let* $\mathcal{T}$ *be a realistic terrain with* n *triangles. Then both the parallel and the perspective visibility map have complexity* $\Theta(n\sqrt{n})$ *in the worst case.*

With the output-sensitive algorithm of Katz, Overmars and Sharir [110], we can compute the visibility map on a realistic terrain in $O(n\sqrt{n}\log n)$ time.

5.4 The shortest path map and the Voronoi diagram

Recall from Section 5.2 that from now on, we have an additional, fourth assumption, on the dihedral angle of the triangles of $\mathcal{T}$. In order to study the complexity of shortest path maps and Voronoi diagrams on realistic terrains, we first need some definitions on shortest paths and bisectors, and results on their complexity.

Shortest paths on terrains were studied extensively in [40, 108, 132]. A geodesic between two points is a path that is locally shortest. An actual shortest path is a geodesic that has minimum length. A geodesic that crosses the interior of an edge between two triangles is such that if the triangles are made coplanar by rotation about their common edge, then the geodesic becomes a straight line segment on the union of the two triangles. The rotation is referred to as *unfolding*. Following [132], a geodesic path "is an alternating sequence of vertices and (possibly empty) edge sequences such that the unfolded image of the path along any edge sequence is a straight line segment, and the angle passing through a vertex is greater than or equal to π." A shortest path has the additional property that no edge can occur more than once. The *root* of a path from p to p' is the last vertex we encounter on a shortest path from p to p'; if there is no such vertex, then p is the root. Recall from Lemma 5.2.8 in Section 5.2.3 that a shortest path between two points on a realistic terrain has complexity $\Theta(\sqrt{n})$.

5.4.1 Bisectors

The bisector $B(p, q)$ of a point p and a point q on $\mathcal{T}$ is the set of points on $\mathcal{T}$ with equal distance to p and q. It is a simple curve (open or closed) that consists of line segments and hyperbolic arcs [132]. Imagine that we walk along $B(p, q)$, while we 'sweep' the shortest path of the current point to p across the terrain. At certain points on $B(p, q)$, we encounter a discontinuity in the sweep; the edge sequence of the shortest path from either p or q to the current point on $B(p, q)$ changes. We call such points on $B(p, q)$ *breakpoints*. Note that breakpoints on the bisector $B(p, q)$ are points that have two shortest paths from p or two shortest paths from q.

The worst-case complexity of a bisector is $\Theta(n^2)$ on general terrains. In the terrain in Figure 5.17(a), a set of $\Theta(n)$ skinny peaks form two fans, which are directed away from both p and q. Along the northwest–southeast diagonal, p and q are alternatingly closest. The bisector $B(p, q)$ has $\Omega(n)$ breakpoints, and between each two consecutive breakpoints, it crosses $\Omega(n)$ edges of the terrain.

on a realistic terrain, we cannot create such a quadratic-size bisector. However, it is possible that a bisector on a realistic terrain crosses $\Omega(\sqrt{n})$ edges before it encounters a breakpoint; see Figure 5.17(b). the terrain is horizontal, i.e., it is parallel to the xy-plane, and its vertices lie on a regular $\sqrt{n} \times \sqrt{n}$ grid. The bisector of p and q, which does not have any breakpoints, is a straight line segment which intersects $\Omega(\sqrt{n})$ triangles.

Figure 5.17(c) shows that a bisector on a realistic terrain has $\Omega(n)$ breakpoints. In fact, it has $\Theta(n)$ breakpoints, since it has at most one breakpoint for every vertex.

Lemma 5.4.1. *A bisector of two points on a realistic terrain intersects $\Theta(\sqrt{n})$ triangles between two breakpoints in the worst case.*

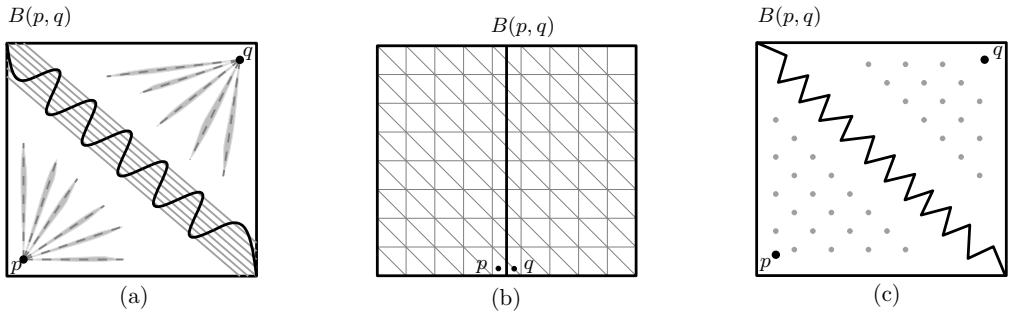


Figure 5.17: (a) On general terrains, the bisector of two points has complexity $\Omega(n^2)$. (b) On a realistic terrain, a bisector can cross $\Omega(\sqrt{n})$ edges between two breakpoints. (c) A bisector on a realistic terrain can have $\Omega(n)$ breakpoints.

Proof: Figure 5.17 shows a lower bound construction; it remains to prove the upper bound. Let b_1 and b_2 be two consecutive breakpoints on $B(p,q)$, and let $P(p,b_1)$ and $P(p,b_2)$ be the shortest paths from p to these two breakpoints, respectively. If there is more than one shortest path to b_1 and/or b_2, choose them so that the area enclosed by $P(p,b_1)$, $P(p,b_2)$, and $B(p,q)$ is smallest. Let r_p be the last common vertex of the shortest paths $P(p,b_1)$ and $P(p,b_2)$, and let r_q be defined analogously; see Figure 5.18.

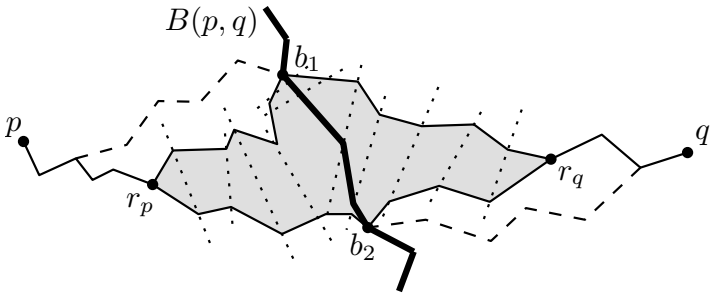


Figure 5.18: Illustration of the proof of Lemma 5.4.1.

By Lemma 5.2.8, all four paths intersect $O(\sqrt{n})$ triangles/edges. Because b_1 and b_2 are consecutive breakpoints on $B(p,q)$, the region on $\mathcal{T}$ that is enclosed by $P(r_p,b_1)$, $P(r_p,b_2)$, $P(r_q,b_1)$, and $P(r_q,b_2)$ does not contain any vertices of $\mathcal{T}$; in Figure 5.18 this region is shaded. Consequently, the edges that intersect $B(p,q)$ between b_1 and b_2 have to intersect, or end on, at least two of the four paths. As we observed above, there are only $O(\sqrt{n})$ such edges, which completes the proof. ∎

We conclude this section with the following, summarizing lemma.

Lemma 5.4.2. *For two points p and q on a realistic terrain $\mathcal{T}$, the bisector $B(p,q)$ has complexity $\Omega(n)$ and $O(n\sqrt{n})$.*

5.4.2 Shortest path maps

The *shortest path map* of a source point s is the subdivision of $\mathcal{T}$ into cells such that the vertex and edge sequence of the shortest path to any point in that cell from the source is the same. For simplicity of argument, we assume that for all vertices of $\mathcal{T}$ the shortest path from s is unique. Obviously, other points on $\mathcal{T}$ can have more than one shortest path.

The analysis of [132] shows that on general terrains, the shortest path map has complexity $\Theta(n)$ on a single triangle, and $\Theta(n^2)$ overall. Furthermore, from the description in [132], it is known that in the shortest path map, every terrain edge is partitioned into $O(n)$ intervals such that the vertex and edge sequence of the shortest paths from the source s to all points within an interval is the same. Any triangle is partitioned into $O(n)$ cells that arise from the additively weighted Voronoi diagram of some set of points; these points are obtained by planar unfoldings of terrain vertices along different edge sequences. The weights are given by the shortest path distances from the source s to these roots.

We conjecture that on a realistic terrain the shortest path map can still have linear complexity on a single triangle. Therefore, it is unlikely that the analysis in [132] will help to obtain a subquadratic complexity bound for realistic terrains. We give a more global analysis that describes where the boundaries of the shortest path map cells originate from to obtain an $O(n\sqrt{n})$ size bound.

Triangle edges necessarily give rise to boundaries of cells in the shortest path map. The cell boundaries that cut through triangles capture whether a shortest path passes to the one side or the other side of a vertex of $\mathcal{T}$, or through the vertex itself. In a terrain, there are two types of vertices: (1) vertices of which the spatial angle, i.e., the sum of the angles of the triangles incident to the vertex, is at least 2π, and (2) vertices with spatial angle less than 2π. From earlier papers on shortest paths on terrains [40, 108, 132], it is known that a shortest path from s to any point on the terrain can only pass through vertices of type (1). These two types of vertices give different boundary types in the shortest path map.

Definition 5.4.1. *The* vertex trace *of a vertex v with total spatial angle $< 2\pi$ is the set of points on the terrain that each have two different shortest paths from s, and such that for any point p on the vertex trace, the two shortest paths from s have the same edge sequences except for edges incident to v.*

Every vertex with spatial angle $< 2\pi$ has one vertex trace. The vertex trace is a single straight line segment on each triangle on which it occurs. It is also the bisector of two different realizations of the same point (some vertex, or s) in the unfolding of the triangles that the shortest paths cross.

Definition 5.4.2. *The* geodesic region *of a vertex v with total spatial angle $\geq 2\pi$ is the region of points on $\mathcal{T}$ such that for any point p in that region, the shortest path from*

the source s to p passes through v as the last vertex. A geodesic trace *of a vertex v with total spatial angle $\geq 2\pi$ is the maximal connected set of points on the boundary of the geodesic region of v that is a subset of a shortest path from s through v.*

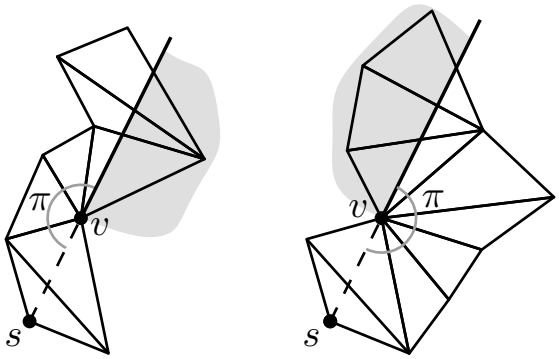


Figure 5.19: Two different unfoldings around v give rise to two geodesic traces (in bold) starting at v, which enclose the geodesic region of v (shaded).

Intuitively, the geodesic traces of a vertex v are the outermost shortest paths that still pass through v. More formally, if the shortest path from s to v leaves v in a certain direction, then the two geodesic traces continue beyond v at angles exactly π from this direction in the unfolding at v. Every vertex with spatial angle $> 2\pi$ has exactly two geodesic traces. If the spatial angle at v is exactly 2π, then the geodesic region degenerates to a linear feature and the geodesic trace is the same as the geodesic region. A geodesic trace is a single straight line segment on each triangle on which it occurs; see Figure 5.19 for an illustration.

We consider traces to be directed away from the source, which means that for a point that moves over the trace in the given direction, it gets further from the source s. Terrain vertices are always starting points of vertex or geodesic traces. The only way in which a trace can end is when it reaches the boundary of the terrain, or when it reaches another trace. The latter case gives rise to junction traces.

Definition 5.4.3. *The* junction trace *of a point q on the terrain, where q is a point where two traces end (any combination of the three types), is the set of points with two different shortest paths from s, and such that the edge sequences of these two shortest paths are the same starting at one edge of the triangle that contains q.*

A junction trace cannot start at a vertex, but starts in the interior of a triangle or on an edge. A junction trace is a single straight line segment or a hyperbolic arc on each triangle on which it occurs (or two or three hyperbolic arcs that are part of the same hyperbola). It is also the bisector of two vertices of the terrain (possibly the same but in different unfoldings), with additive weights. The three different types of traces are illustrated in Figure 5.20.

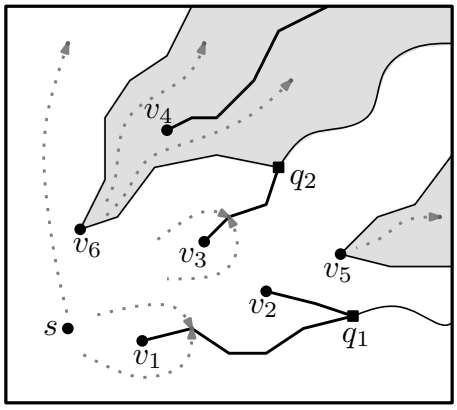


Figure 5.20: Different traces on a terrain: vertices v_1, v_2, v_3, and v_4 give rise to vertex traces, v_5 and v_6 to geodesic regions and corresponding traces, and q_1 and q_2 are starting points of junction traces. The dotted arrows show the directions of shortest paths from s.

On a convex polytope, Aronov and O'Rourke [7] define the ridge tree as the set of points that have two or more distinct paths from the source. A convex polytope or terrain has no geodesic traces, and the vertex traces and junction traces together are the same as the ridge tree.

Lemma 5.4.3. *No pair of traces intersects. No trace properly intersects a shortest path from s.*

Proof: The first part is true by construction. Geodesic traces are parts of a shortest path from s, therefore another shortest path from s cannot intersect a geodesic trace properly.

Vertex traces and junction traces are bisectors of two vertices v and v' (after unfolding), possibly with additive weights. A proper intersection of a shortest path σ with such a trace τ implies that one point on σ has its shortest path from s via v, whereas a point just across τ on σ has its shortest path from s via v'. This contradicts the fact that σ is (part of) a shortest path from s. ■

Lemma 5.4.4. *There are $O(n)$ traces on a terrain with n triangles.*

Proof: There are $O(n)$ vertex and geodesic traces. At a junction trace, two other traces always end and one begins. Hence there are at most $O(n)$ junction traces as well. ■

Lemma 5.4.5. *Any trace has complexity $O(\sqrt{n})$.*

Proof: A geodesic trace is part of a shortest path from s, so by Lemma 5.2.7 it has complexity $O(\sqrt{n})$.

Vertex and junction traces are bisectors of two roots r_1 and r_2 in different unfoldings. If we consider the two endpoints p and q of some trace τ, then there are two shortest paths from s to p, one via root r_1 and one via root r_2. The same is true for q. This implies that the trace τ is a straight line segment or hyperbolic arc in the unfolding of the triangles on which it occurs, and hence it has no breakpoints. Therefore, by the proof of Lemma 5.4.1, each vertex or junction trace has complexity $O(\sqrt{n})$. ■

Lemma 5.4.6. *The subdivision induced by the traces and the triangles is the shortest path map.*

Proof: The proof consists of two parts: we start by showing that two points p and q in the same cell of the subdivision induced by the traces and triangles have the same edge sequence from s, then we show the converse.

Let p and q lie in the same cell of the subdivision induced by the edges of T and the traces. Clearly p and q must be in the same triangle of T. Consider the shortest paths from s to p and q and the corresponding edge sequences. Assume for a contradiction that the edge sequences differ, and consider the triangle t closest to p and q where this occurs. The shortest paths from s to p and q leave t through the same edge, but do not enter it through the same edge.

First, assume that the shortest paths enter t through different edges, and let the shared endpoint of these edges be v. This case is illustrated by the left part of Figure 5.21. Then v has a trace, or possibly two. By Lemma 5.4.3, traces and shortest paths do not intersect, so the trace of v must be between the shortest paths of p and q on all triangles intersected by the shortest paths between t and p and q. Hence, p and q cannot be in the same cell, a contradiction with the assumption that they have a different edge sequence.

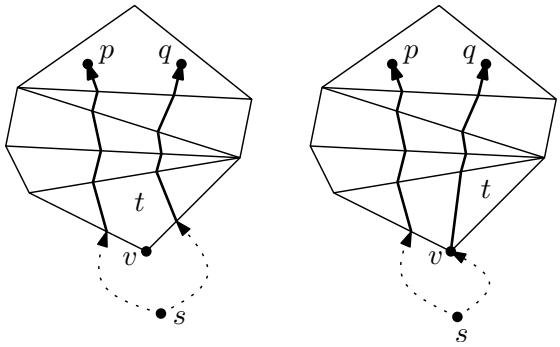


Figure 5.21: Illustration of the proof of Lemma 5.4.6.

Second, assume that one shortest path (say, to p) enters t through an edge and the other shortest path (to q) enters t through a vertex; this vertex v is opposite to the edge

through which both shortest paths leave t; see the right of Figure 5.21 for an illustration. The total spatial angle at v is at least 2π, so v has one or two geodesic traces. By the angle properties at v, we have two cases: (i) the shortest path to q lies between the two geodesic traces of v, which is a contradiction to the fact that p and q lie in the same cell of the subdivison, or (ii) the shortest path to q coincides partially with (one of the) geodesic trace(s) of v. In the latter case, we distinguish three subcases: (a) the other geodesic trace lies strictly between the two shortest paths to p and q, in which case p and q cannot lie in the same cell of the subdivision, which is a contradiction again, (b) q itself lies on a geodesic trace of v, again a contradiction to the fact that p and q are in the same cell, or (c) a junction trace starts at some point z on the shortest path between v and q. Assuming z lies in the interior of a triangle, the shortest path of q goes straight through z, whereas the three traces have angles less than π between them (for hyperbolic arcs we refer to the angle of the tangent at z). This is true because on a triangle, the traces are the additively weighted Voronoi diagram of some set of sites. Hence, locally at z, either the other trace arriving at z or the junction trace leaving z is between the shortest paths of p and q. Since traces do not intersect shortest paths, both cases yield that p and q do not lie in the same cell. Note that the proof arguments also apply to the case where t is the triangle that contains p and q.

In the second part of this proof, we show that the converse is also true, i.e., two points p and q that have the same edge sequence from s lie in the same cell of the subdivision induced by the traces and triangles.

Assume for a contradiction that p and q do not lie in the same cell of the subdivision. If p and q lie in different triangles of $\mathcal{T}$, then they clearly have different edge sequences from s and we are done. Otherwise, they lie in the same triangle t of $\mathcal{T}$ and there is a trace τ that separates p and q on t. We only consider the case that p and q lie in adjacents cells that share τ as a common boundary; the proof in the other cases proceeds similarly. Since the shortest path from s to p does not intersect any trace, it must stay on the other side of τ than the shortest path to q. If τ is a vertex or geodesic trace induced by vertex v, then the shortest paths from s to p and q pass on different sides of v, and thus these paths have different edge sequences. If τ is a junction trace, then there exists at least two vertices v and w, which (indirectly) induce τ, such that the shortest paths from s to p and q pass on opposite sides of both v and w, and thus they have different edge sequences, which completes the proof. ∎

Theorem 5.4.1. *The shortest path map of a realistic terrain has complexity* $\Theta(n\sqrt{n})$ *in the worst case.*

Proof: The upper bound directly follows from the lemmas above. For the lower bound, take a convex terrain so that there are no geodesic traces. If we place the vertices on a nearly flat surface, then vertex traces never merge into junction traces. At least $\Omega(n)$ vertex traces will intersect $\Omega(\sqrt{n})$ edges of $\mathcal{T}$. ∎

Although we could not use the combinatorial analysis of Mitchell et al. [132] to obtain better complexity bounds, it is easy to verify that the total number of points for

which their algorithm computes the additively weighted Voronoi diagram is $O(n\sqrt{n})$ on a realistic terrain, and therefore the shortest path map for a point on a realistic terrain can be computed in $O(n\sqrt{n}\log n)$ time.

5.4.3 Voronoi diagram

Let S be a set of m sites on a realistic terrain $\mathcal{T}$. We are interested in the maximum complexity of the Voronoi diagram of S on $\mathcal{T}$, denoted by $\mathrm{VD}(S,\mathcal{T})$, where distances are shortest path distances on $\mathcal{T}$. Each of the m Voronoi cells, for the sites $s_i \in S$, on $\mathcal{T}$ is connected, but not necessarily simply-connected. By Euler's formula this implies that there are $O(m)$ Voronoi vertices of degree 3 or higher. Furthermore, only $O(m)$ bisectors appear in the Voronoi diagram of S on $\mathcal{T}$. Together with Lemma 5.4.2, this immediately leads to an $O(mn\sqrt{n})$ bound on the complexity, but we will show that it is $O((n+m)\sqrt{n})$. First we give a lower bound of $\Omega(n+m\sqrt{n})$.

Lemma 5.4.7. *The Voronoi diagram of m sites on a realistic terrain has complexity $\Omega(n+m\sqrt{n})$.*

Proof: If the $\Omega(n)$ term dominates the $\Omega(m\sqrt{n})$ term, we simply use two sites that give a bisector of complexity $\Omega(n)$. Otherwise, we place m points on a flat, horizontal terrain in a row, so that they have the same y-coordinates. If we place the terrain vertices on a regular grid, then each of the $m-1$ bisectors intersects $\Omega(\sqrt{n})$ triangles between the boundaries with the terrain. ■

As we mentioned before, the Voronoi diagram $\mathrm{VD}(S,\mathcal{T})$ has $O(m)$ vertices of degree 3 (or higher), and consists of parts of bisectors that intersect edges and contain breakpoints. Since any vertex can only give a breakpoint in a bisector of a site in whose cell that vertex lies, we can prove the following lemma.

Lemma 5.4.8. *The total number of breakpoints in $\mathrm{VD}(S,\mathcal{T})$ is $O(n+m)$.*

Proof: Every vertex of $\mathcal{T}$ lies in exactly one Voronoi cell; let v be some vertex and assume it lies in the cell c_i of site s_i. We consider only those traces that start at v and are created by source s_i. Vertex v may have one vertex trace or two geodesic traces incident to it. By Lemma 5.4.4, there are $O(n_i)$ such traces inside c_i, where n_i is the number of vertices inside c_i. We will show that the number of intersections of traces with the boundary of c_i is $O(n_i+h_i)$, where h_i is the number of Voronoi cells that lie in the interior of c_i, and share at least one Voronoi edge with c_i. These intersections are exactly the breakpoints of the bisectors associated with s_i that occur in $\mathrm{VD}(S,\mathcal{T})$. We have two cases: (i) the Voronoi cell c_i does not contain any other Voronoi cells in its interior, and (ii) c_i does contain other cells as 'holes'. We analyse these two cases separately. We only consider vertex traces; the proof is analogous for the case that v gives rise to a geodesic trace, or the trace starting at v has merged with another trace into a junction trace before reaching the boundary of c_i.

For the first case, assume the vertex trace of v reaches the boundary of cell c_i at a point p, disappears when it leaves c_i, and reappears in c_i at another point p'; see

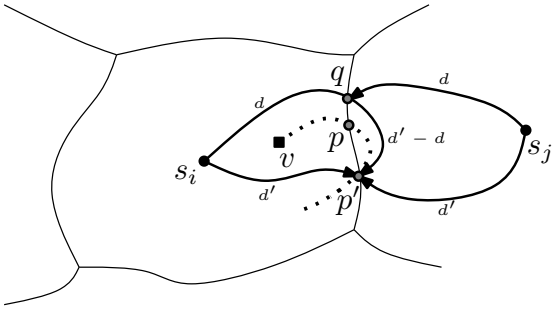


Figure 5.22: The vertex trace of v cannot reenter the Voronoi cell c_i.

Figure 5.22. Then at p', there are two equally long shortest paths from s_i and one also equally long shortest path from another site s_j. The two shortest paths from s_i enclose the vertex v and they do not intersect the trace of v. Therefore, one of the shortest paths from s_i must contain points strictly in the interior of the cell c_j. Assume one shortest path to p' enters c_i at a point q. If the distance from s_i and s_j to q is d and the distance from to s_i and s_j to p' is d', then the distance from p' to q is $d' - d$. Consequently, there is also a second shortest path from s_j to p', and the whole shortest path between p' and q has equal distance from s_i and s_j. But then it is part of the bisector, which contradicts the fact that the path from q to p' lies inside c_j. This concludes the proof for case (i).

In the second case, let c_j be the Voronoi cell of a site s_j which lies in the interior of c_i. Again, let v be a vertex in c_i, and assume that the vertex trace τ_v of v reaches the boundary between cell c_i and c_j at a point p, disappears, and reappears in c_i at another point p'. This time, we cannot argue that one of the shortest paths from p' to s_i must contain points strictly in the interior of the cell c_j, since these paths can enclose c_j, and indeed, it is possible that the trace τ_v reappears in c_i. However, we can show that this can only happen once for every cell c_j inside c_i. For a contradiction, assume that another trace τ_u also reappears in c_i, at a point q; see Figure 5.23.

By the argumentation for the first case, the two shortest paths from s to q cannot go through c_j, which implies that one of these paths must intersect some trace, which is not possible by Lemma 5.4.3. Thus, there is only one 'reappearing trace' per Voronoi cell that lies within c_i. Therefore, in case (ii), the number of intersections of traces with the cell boundary is $O(n_i + h_i)$.

Finally, we observe that the total number of breakpoints is $\sum_{i=1}^{m}(n_i + h_i)$, which is $O(n + m)$. ∎

Between the $O(m)$ Voronoi vertices and the $O(n + m)$ breakpoints, the bisectors have complexity $O(\sqrt{n})$ as shown in Lemma 5.4.2, and hence we conclude with the following theorem.

Theorem 5.4.2. *The Voronoi diagram of a set of m sites on a realistic terrain with n*

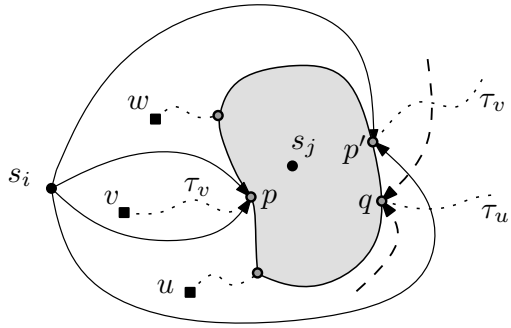


Figure 5.23: The Voronoi cell c_j is contained in the cell c_i.

triangles has complexity $\Omega(n + m\sqrt{n})$ *and* $O((n + m)\sqrt{n})$ *in the worst case.*

As in the previous section, the algorithm of Mitchell et al. [132] can be shown to handle less events on realistic terrains, and thus that it computes the Voronoi diagram of m sites in $O((n + m)\sqrt{n}\log n)$ time.

5.5 Concluding remarks

This chapter studied realistic input models for polyhedral terrains, a topic that has not been considered so far. We have made three input assumptions that together are sufficient to show a subquadratic upper bound on the complexity of the visibility map. Furthermore, we have shown that no subset of our assumptions is sufficient to achieve the same. For shortest paths, bisectors, shortest path maps, and Voronoi diagrams on terrains, we have used a fourth input assumption and proved upper and lower bounds on their complexities. Our research helps to explain the discrepancy between the worst-case performance of algorithms on polyhedral terrains and their efficiency in practice.

The upper and lower bounds for visibility maps, shortest paths, and the shortest path map are tight, but there is a considerable gap for bisectors and Voronoi diagrams. This is the most important open problem that arises from this chapter.

Most of our bounds are $O(n\sqrt{n})$ for realistic terrains, in contrast with quadratic for general terrains. However, in practice we may expect even lower bounds for the visibility map, like close to linear. The question is whether a (slightly) stronger set of realistic assumptions exists that leads to such a bound. On the other hand, this study is a first attempt at defining a realistic input model for polyhedral terrains, and therefore, it would be interesting to replace one (or more) of our assumptions with a weaker version, while still obtaining the same bounds.

An approach that may be worth studying in this context is that of smoothed analysis [179, 180], in which one measures the complexity of algorithms and data structures under the assumption that the input is subject to small amounts of random noise. It is

conceivable that by a slight perturbation of the vertex locations of the terrain, the probability that a worst-case construction occurs is reduced significantly, and thus one would be able to show improved (expected) upper bounds on the combinatorial complexity of the various structures discussed in this chapter.

Chapter 6

Experimental verification of a realistic input model for polyhedral terrains

The contents of this chapter have been published as:

E. Moet.
Experimental verification of a realistic input model for polyhedral terrains.
Technical Report UU-CS-2007-052, Universiteit Utrecht, December 2007.

Abstract

A realistic input model for polyhedral terrains was introduced in Chapter 5. With this model, improved upper and lower bounds on the combinatorial complexity of visibility and distance structures on these terrains were obtained. In this chapter, we perform experiments with real-world data to determine how realistic the assumptions of that input model are, by examining what the values of the model parameters are in practice. We consider the results of these experiments as evidence that the assumptions of the model are probably realistic: they are all satisfied by the terrains that we generated from GIS data. Moreover, we evaluate the complexity of the visibility map in these terrains by counting vertices of the visibility maps for different viewpoints, thus collecting evidence that it may be unlikely to find a visibility map that has significantly higher complexity than the bounds in Chapter 5.

6.1 Introduction

For many geometric problems, the theoretically proven worst-case computation and combinatorial complexity bounds are higher than the running time and storage that are observed in practical applications. Realistic input models are a way to explain this discrepancy between theory and practice. By describing certain criteria which the input has to satisfy, it is possible to show better bounds; these improved bounds are usually closer, or even equal, to the real-world complexities.

Polyhedral terrains are used extensively in GIS, computer graphics, and computational geometry, since they provide a way to approximate parts of the surface of the earth. In principle, they are triangulations in which each vertex has a height coordinate associated with it. Since for every x- and y-coordinate, there is only one z-coordinate such that the point (x, y, z) belongs to the terrain, they are sometimes considered to be 2.5-dimensional.

Many well-known two- and three-dimensional data structures can also be defined for the special case that a polyhedral terrain is the input, where the metric is the distance travelled over the (triangles of the) terrain. For example, one can consider the Voronoi diagram of a set of points on a terrain, or the shortest path map on a terrain for a single source, or even for multiple sources. For GIS applications, the visibility map has been studied extensively. It is the three-dimensional variant of the two-dimensional visibility polygon or diagram. Since a polyhedral terrain can be considered as a hybrid between 2D and 3D, we expect the complexities of algorithms and data structures for terrains to be worse than in 2D but better than in 3D. However, for the visibility map for instance, the worst-case complexity bound is the same for a terrain as it is for general 3D scenes, namely $\Theta(n^2)$, where n is the number of triangles of the terrain. In spite of that, the worst-case scenario hardly ever occurs in practical situations, and thus the complexity that is observed in applications rarely is (close to) quadratic. Therefore, the discrepancy between theory and practice also exists for polyhedral terrains.

In Chapter 5, a realistic input model for polyhedral terrains was introduced. It put forward several assumptions that allowed for improved, and supposedly more realistic,

complexity bounds for three different structures on terrains. The goal of this chapter is to check to what extent these assumptions, and the bounds they lead to, are indeed realistic.

We perform several experiments to determine how realistic the assumptions of the input model from Chapter 5 are. The values of the four input model parameters α, β, c, and d are evaluated for different test scenes of varying size (i.e., number n of triangles), representing landscapes in the US. We observe that these experimental values do not depend on n, which is what is assumed in the input model. This is evidence that the assumptions are indeed realistic. In our experiments, we also determine the complexity of the visibility map for different viewpoints in the test scenes to observe what this complexity is in practice. We view these empirical results as evidence that it may be unlikely to find a visibility map that has significantly higher complexity than the bounds in Chapter 5..

The structure of this chapter is as follows. In Section 6.2, we give details on the setup of our experiments. Then we repeat the input model assumptions. In Section 6.4, for each of the input model parameter, we discuss the method of computation and present the results we have obtained. Finally, in Section 6.4.4, we present the complexity of the visibility map that we observed in practice.

6.2 Experimental setup

Verifying whether an input model is realistic starts with *reality*, that is, input data from practice. In this case, polyhedral terrains are supposed to model landscapes, or in other words, parts of the earth. The real-life data comes from Geographical Information Systems (GIS). In this section, we give more details of the raw data that we used, how we acquired it, and how we preprocessed it to use in our experiments.

6.2.1 DEM data

In *remote sensing*, the elevation of the surface is measured at sample points that lie on a regular grid, which yields a (two-dimensional) matrix with height values. Such a matrix is usually called a Digital Elevation Model (DEM). The DEM-data that we use in our experiments was downloaded for free from `http://www.mapmart.com/`, the GIS website maintained by IntraSearch Inc. The data was originally collected by the U.S. Geological Survey (USGS). The areas that we analyze all lie in either Colorado or Nevada; the data set consists of 16 scenes and contains both urban and mountainous landscapes.

The downloaded data is set in the Spatial Data Transfer Standard format (SDTS). The cellsize in all DEMs is 30 meter. We convert these SDTS DEMs to the more intuitive and compact ARC/info ASCII Grid format; several utilities that provide this procedure can be downloaded from the web. The syntax of the header of this file format is shown in Table 6.1; every line starts with a (textual) identifier, followed by a numerical value corresponding to that identifier. After this header, the actual grid data is given, in **nrow** rows of **ncol** numerical (height) values. The height value is missing

for some coordinates; there is an indicator for these grid points, which usually is the value -9999.

`ncols` ncol	(Number of columns in the grid)
`nrows` nrow	(Number of rows in the grid)
`xllcorner` x	(Lower left x-coordinate of the grid)
`yllcorner` y	(Lower left y-coordinate of the grid)
`cellsize` size	(Grid cell size)
`NODATA_value` NODATA	(Value indicating an empty grid cell)

Table 6.1: The header of the ARC/info ASCII Grid file format.

Each DEM consists of around 160,000 height values, corresponding to a grid of approximately 360 rows and 470 columns. If we let the rows correspond with x-coordinates and the columns with y-coordinates, then the height values are the corresponding z-values. While reading the input data we determine the range of z-coordinates, to be able to scale all 3D-coordinates to smaller values that fall into the range of screen-coordinates. This so-called z-range also gives an indication of how mountainous the scene is. The characteristics of all sixteen DEM files are given in Table 6.2.

Name	State	#rows	#cols	#points	z-range
Antelope Peak	NV	362	465	168330	[1887,3184] (1297)
Arvada	CO	357	462	164934	[1572,1729] (157)
Aspen	CO	370	470	166494	[7415,12218] (4803)
Castle Peak	CO	367	470	165022	[2024, 3440] (1416)
Coffin Mountain	NV	359	466	163697	[1577, 2486] (909)
Commerce City	CO	357	462	164927	[1536, 1667] (131)
Crooked Creek	CO	368	469	165892	[2484, 3672] (1188)
Eagle	CO	366	469	165269	[1959, 3196] (1237)
Golden	CO	358	468	165016	[5389, 7070] (1681)
Grouse Mountain	CO	366	469	165576	[2268, 3977] (1709)
Lava Creek	CO	365	468	164966	[2119,3502] (1384)
Mount of the Holy Cross	CO	366	468	165857	[2660, 4266] (1606)
Mount Jackson	CO	366	468	165821	[2870,4152] (1293)
Red Cliff	CO	365	467	165569	[2611, 3651] (1040)
Red Creek	CO	367	469	165857	[2207, 3664] (1457)
Suicide Mountain	CO	367	470	165601	[1962, 3418] (1456)

Table 6.2: DEM characteristics for the 16 scenes which are used in the experiments.

6.2.2 TIN generation

To obtain a continuous input data set from the discrete input, it is customary in GIS to construct a Triangular Irregular Network (TIN) that represents such a DEM as

good as possible; see Figure 6.1. The elevations at (some of the) grid points give the vertices of the TIN, and the triangles give interpolated height values at all other x- and y-coordinates. A TIN is the same thing as what is called a polyhedral terrain in computational geometry. In this section, we describe how we obtain polyhedral terrains from the input data. We evaluate the input model parameters for these generated terrains in Section 6.4.

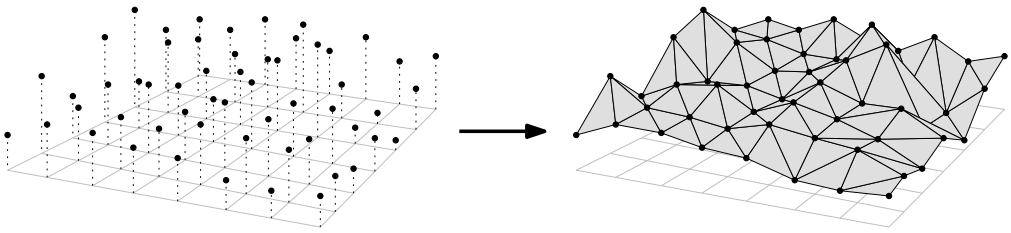


Figure 6.1: A DEM (left) and a TIN that represents this DEM (right).

Besides the fact that elevation data is often missing at a number of grid points, it is common that not all sample points of the DEM are inserted as vertices into the TIN, as is the case in Figure 6.1. TINs generally are denser in areas where the elevation is more variable, and sparser in flat areas. Thus, TINs help to reduce storage space and computation time while they still preserve the most important characteristics of the input. Obviously, by selecting a subset of the DEM grid points, some data loss occurs, but this is usually acceptable since DEMs are already approximations of the real world.

Related work

There are many different approaches to construct a triangulated irregular network (TIN) from a digital elevation model (DEM). Two groups of methods can be distinguished: namely, *surface simplification* and *surface refinement*.

Algorithms in the first group start with the original surface and simplify it until the number of vertices is small enough, while the error of approximation is still acceptable. This type of method inherently also provides a multi-resolution hierarchy of approximating surfaces. Most succesfull simplification methods are based on *iterative edge contraction* [106, 128], or the contraction of (not necessarily incident) vertex pairs [84]. Another simplification method is the so-called *drop heuristic* [125], in which DEM points are successively removed, based on significance, until a certain error level is reached. More details on the various polygonal surface simplification algorithms can be found in two surveys [100, 129].

Surface refinement algorithms start with a very coarse surface and then increase the resolution until the desired level of approximation is obtained. These refinement methods are sometimes called *iterative vertex insertion methods*, and vary based on the selection criterium that determines which DEM grid points are inserted as vertices of the TIN. As the name suggest, iterative vertex insertion methods work iteratively, which means that while a subset of the vertices is being selected, a triangulation (of the

convex hull) of the selected vertices so far is maintained. The most common choice is the Delaunay triangulation because of several desirable properties, such as maximizing the minimum angle. One of the first and quite succesfull methods was presented by Fowler and Little [76]. They describe a global procedure that identifies features of the surface such as peaks and ridges, and selects the vertices on these features for insertion into the TIN. The very important point (VIP) method is another well-known vertex selection method. It is a local procedure that determines the significance of a point relative to its 8-connected neighborhood [39]. De Berg et al. use this method in their work on realistic input models for geometric algorithms [22].

Our method

We use a different, intuitive approach called the *greedy insertion method*, which is a surface refinement algorithm, and which has been described in a number of different fields, such as GIS [102], computer graphics [83], computer-aided design (CAD) [72], and computational geometry [71]. The vertex that is inserted in each step, is the grid point that defines the maximum approximation error, i.e., the DEM element that has the greatest deviation from the current approximating terrain. Originally, one would iterate until a given level of error was reached, but since we want to obtain a TIN with a specific number of triangles, we iterate until that given size is reached.

For each scene, we create thirteen TINs with increasing *resolution*, i.e., a growing number of triangles n. We let n range over [500,20000]. For completeness, we also create the TIN with all DEM grid points; we call this TIN the *full terrain*. The data structure that we use to store the TIN is a two-dimensional Delaunay triangulation, which is implemented by `CGAL` [37]. We adapt this data structure in a straightforward way so that every vertex of the 2D triangulation stores a height value. In this way, the projection of the TIN is always a Delaunay triangulation. When we insert a new (3D) grid point into the terrain, we insert the xy-projection of this point into the (2D) triangulation, while storing the z-coordinate in the resulting vertex, after which the `CGAL` implementation ensures that the (2D) Delaunay property is automatically restored.

The first step in the TIN generation algorithm is the initialization of the TIN. At the boundary of the DEM, input data may be missing for some grid points. These grid cells are flagged by the `NODATA`-indicator. Since we want to create a rectangular TIN, which is common in GIS, we find a (maximal) subgrid such that the four corner points all do contain elevation data. We insert these four corner points into an initially empty Delaunay triangulation.

For a given value n, we now want to create a TIN that consists of n triangles, by inserting the *most deviating* grid points iteratively. Which grid point deviates the most from the 'current' terrain is determined in the following way. For each grid point (x, y, z), we locate the triangle that lies above resp. below it. For this triangle, we compute the supporting plane, which we use to compute the interpolated height value z' for (x, y). We compute the deviation $|z' - z|$, and find the maximum deviation over all grid points. This point is inserted as a new vertex into the Delaunay triangulation. This process is repeated until the number of triangles of the TIN is at least n (it can be slightly more). In Figure 6.2, four resulting TINs for the scene Antelope Peak (NV)

are shown, for the values $n = 1000, 4000, 12000, 20000$. In Appendix 6.5, the TINs with the same values of n are displayed for the remaining fifteen scenes.

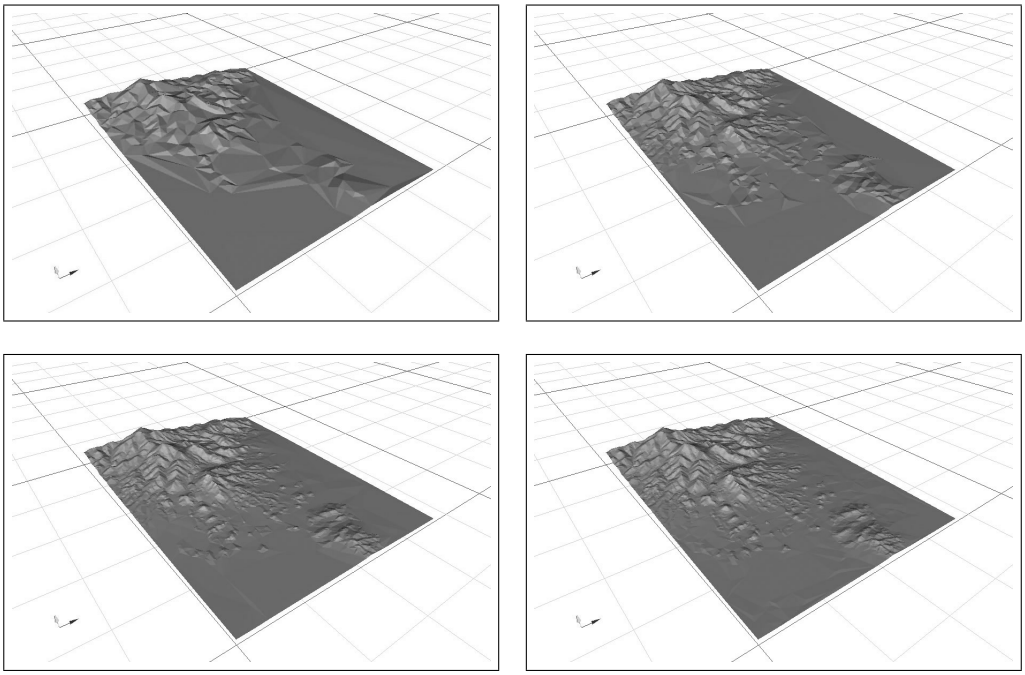


Figure 6.2: Antelope Peak (NV).

Running time analysis

Let N be the number of grid points, and let n be the number of triangles in the resulting TIN. Filling a two-dimensional grid with the DEM values and scaling all coordinates takes $O(N)$ time. The initialization takes $O(N)$ time in the worst case, but will on average take $O(1)$ time. We do not know how efficiently the point location method of the `CGAL`-Delaunay triangulation data structure is implemented, but in the worst case it takes $O(n_i)$ time, if the (intermediate) TIN has n_i triangles. Thus, the time complexity for inserting vertices until the TIN has n triangles is at most $\sum_{4 \le n_i \le n} O(n_i N)$. This yields an overall time complexity of $O(n^2 N)$. The whole TIN generation algorithm, and in particular the point location method, can be implemented in a much smarter, and thus more efficient, way. A typical running time for an implementation of the greedy vertex insertion algorithm is $O(Nn)$ or $O(N \log n + n^2)$, whereas the optimized algorithm by Garland and Heckbert typically takes $O((N+n) \log n)$ time [83]. However, since in this context TIN generation is only a preprocessing step, we did not attempt to optimize our implementation.

6.3 The model

We use the same definitions and notation as in Chapter 5. Let $\mathcal{T}$ be a polyhedral terrain: a piecewise-linear continuous function defined over the triangles of a triangulation in the xy-plane. A terrain $\mathcal{T}$ comprises a set T of n triangles, a set E of n_e edges, and a set V of n_v vertices, where n_e and n_v are $O(n)$.

The four assumptions of the input model are given below; assumptions 1 to 3 refer to the xy-projections of T and E, whereas the last assumption refers to the original triangles in $\mathbb{R}^3$. The restriction that we put on the values α, c, d, and β in the assumptions below is that they all are positive constants, i.e., constants strictly greater than zero.

1. **Fatness**: the minimum angle of any triangle in T is at least α.
2. **Aspect ratio**: the smallest rectangle that contains T has side lengths 1 and c.
3. **Edge length ratio**: the longest edge in E is at most d times as long as the shortest one.
4. **Dihedral angle**: the dihedral angle of the supporting plane of any triangle in T with the xy-plane is at most β, where $\beta < \frac{\pi}{2}$.

In Chapter 5, the fatness assumption is replaced by low-density. Fatness is a stronger restriction than low-density, i.e., the number of terrains that satisfy low-density is larger than those that have only fat triangles. In theoretical respect, we prefer to work with low-density, since this is a more general assumption and thus yields stronger results. However, in this empirical context, we study fatness. Terrains with fat triangles also obey the low-density property. In particular, when you show that for terrains in practice, fatness is a realistic assumption, this implies that low-density also is realistic.

6.4 Evaluation of the model parameters

In this section, we determine the values of all four model parameters α, β, c, and d for all sixteen input scenes, and evaluate whether or not the assumption that they are all constants is realistic.

6.4.1 Aspect ratio

The parameter c can be considered as the aspect ratio of the terrain domain. We treat this parameter separately and briefly, since it differs significantly in nature from the other ones. That is to say, the aspect ratio parameter refers to the terrain as a single entity, whereas the other three model parameters view the terrain as a set of triangles.

The aspect ratio of the terrain is equal to the aspect ratio of the input domain. This ratio depends on the number of rows and columns of the grid that contains the DEM data, and is therefore independent of the number of triangles n, which is chosen later on. Of course, this ratio can change when we select a subgrid of the original DEM to construct the TIN from. Still, for all our scenes, the parameter c is a constant with

respect to n. In our experiments, all values of c lie in the range $[1.287, 1.315]$. All aspect ratios of our sixteen scenes are given in Table 6.3. First, we give the aspect ratio of the full DEM, and then the aspect ratio of the selected subgrid out of which the TINs are generated. Since this parameter is equal for all values of n, the second assumption of the input model from Chapter 5 is satisfied by all our terrains from practice.

Scene	State	DEM a.r.	subgrid a.r.
Antelope Peak	NV	1.285	1.292
Arvada	CO	1.295	1.296
Aspen	CO	1.271	1.287
Castle Peak	CO	1.281	1.298
Coffin Mountain	NV	1.299	1.308
Commerce City	CO	1.295	1.297
Crooked Creek	CO	1.275	1.289
Eagle	CO	1.282	1.298
Golden	CO	1.304	1.315
Grouse Mountain	CO	1.282	1.296
Lava Creek	CO	1.283	1.297
Mount of the Holy Cross	CO	1.279	1.292
Mount Jackson	CO	1.279	1.293
Red Cliff	CO	1.280	1.291
Red Creek	CO	1.279	1.294
Suicide Mountain	CO	1.281	1.297

Table 6.3: The aspect ratio of the 16 scenes.

For the other three parameters α, β and d, the number of triangles can be of influence, so we treat them in more detail in the sections below.

6.4.2 Fatness

In this section, we determine the value of the fatness of triangulations in practice.

We refer to the xy-projection of the terrain, i.e., to a two-dimensional triangulation. For every triangle, we compute the angle at each of the three vertices. We take the two (normalized) vectors, pointing out of the vertex along the two incident edges, and take the arccosine of their dot product. The minimum of the three angles gives the fatness of the triangle. Finally, we compute the minimum over all triangles; this is the fatness of the triangulation. The input model states that this quantity should be bounded from below by a constant $\alpha > 0$.

Results

In Figure 6.3, the graphs with the results of our experiments are displayed. We present four graphs; in each graph we group the results of four scenes. Recall that for each scene, we have thirteen terrains, with a growing number of triangles n. Thus, we can

set out the fatness values for these terrains against n. Since n is much larger for the full terrain, the values for these terrains are not included in these graphs. The fatness of each of these full terrains lies in the range $[2.796, 18.294]$ with an average of 15.358.

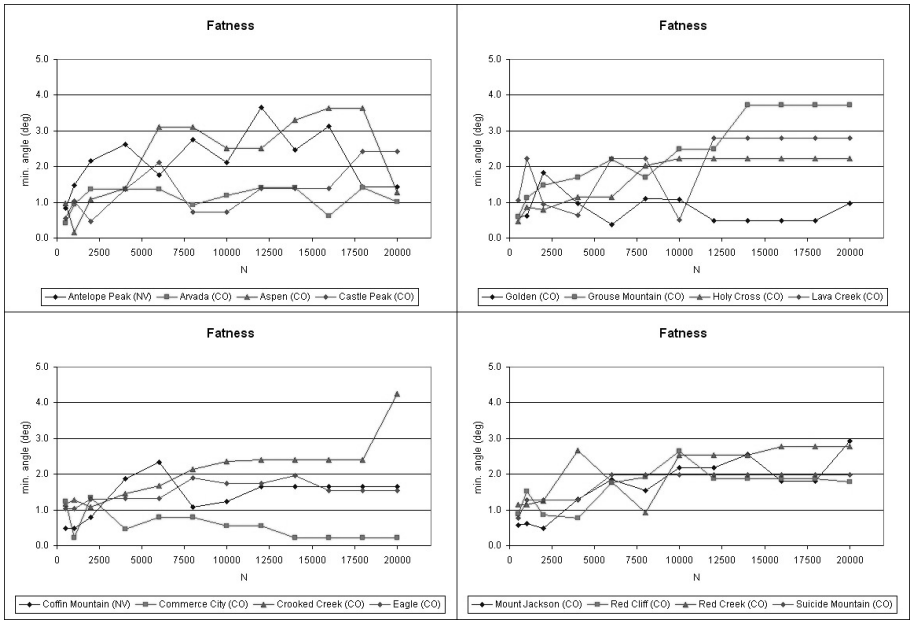


Figure 6.3: Fatness as a function of n.

Conclusion

The fatness of a triangulation of the vertices of a entirely filled regular grid is 45 degrees; all triangles in such a triangulation are right isosceles triangles with side lenghts 1, 1, and $\sqrt{2}$. In practice some input data will be missing, and thus this optimal 45 degrees will generally not be attained.

The input assumption states that the minimum angle over all triangles should never be arbitrarily close to zero. It seems this will not the case. In the majority of the graphs the fatness remains the same when n grows, or it even increases. For five scenes, the main tendency of the graph is not increasing: Antelope Peak (NV), Arvada (CO), Aspen (CO), Commerce City (CO), and Red Cliff (CO). But even for these scenes, the fatness for the maximum n (i.e., the full terrains) is still bounded away from zero by a constant $\alpha > 0$, namely 18.294, 15.101, 18.225, 15.101, and 2.796 respectively.

We observe that, for our input set, the smallest angle in a TIN does not become arbitrarily small if n becomes very large. This means that the first assumption of the input model of Chapter 5 is satisfied in our terrains from practice, and thus we call it realistic for this input set.

6.4.3 Edge length ratio

Next, we evaluate which values are realistic for the ratio between the length of the longest and the shortest edge of a triangulation.

Computing the edge length ratio of a triangulation is very straightforward; you iterate over all edges, and maintain the maximum and the minimum length that you encounter. Finally, you divide these two quantities. The input model states that this ratio should be bounded from above by a constant $d > 0$.

Results

In Figure 6.4, the graphs with the results of our experiments are displayed. As in the fatness graphs, the full terrains have not been included here. The edge length ratios for these full terrains lie in the range $[3.171, 8.000]$ with an average of 4.363.

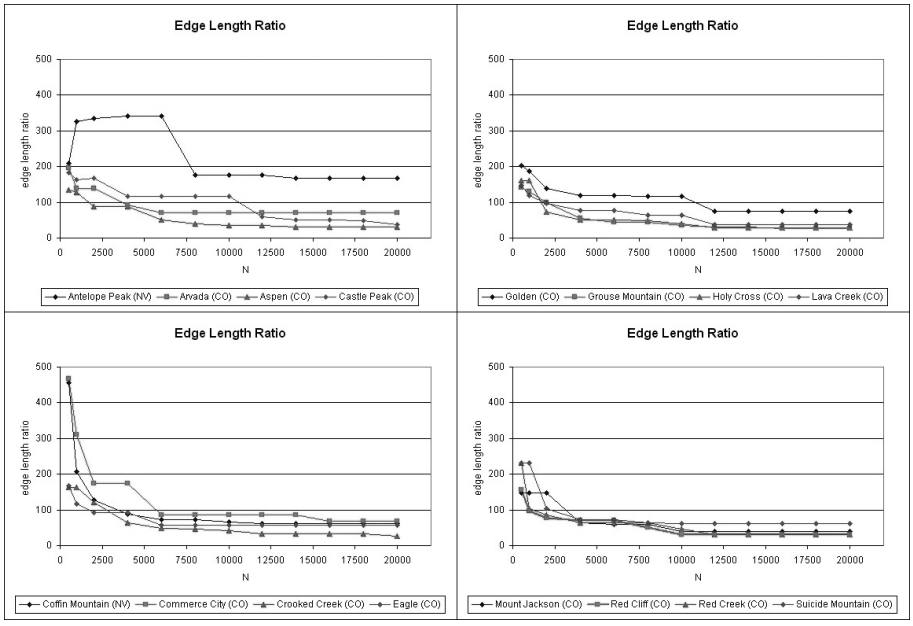


Figure 6.4: Edge length ratio as a function of n.

Conclusion

In a triangulation of the vertices of a entirely filled regular grid, the maximum edge length is $\sqrt{2}$ and the minimum is 1, which yields an edge length ratio of $\sqrt{2}$. This optimal value is the same for *all* scenes created from an entirely filled DEM input.

If we set out the edge length ratio d against n, we see that, generally, d decreases if n increases. For most scenes, the rate at which d decreases initially is high, but

becomes smaller for larger n. Note that even though the edge length ratio itself is not a constant with respect to n, it is (asymptotically) bounded from above by a constant d, which is exactly what the third input model assumption states. Note that grid data may be missing from a DEM and thus the edge length ratio will most likely not attain its optimal value of $\sqrt{2}$. Hence, we observe that the third input model assumption is satisfied by our terrains as well, and we call it realistic for this input set.

6.4.4 Dihedral angle

Finally, we determine whether or not the fourth assumption of the input model of Chapter 5 is satisfied by terrains from practice.

For every triangle of the TIN, we compute the supporting plane; this is the plane of which we need to compute the dihedral angle it makes with the xy-plane. Then we compute a unit normal vector for this plane. To compute the dihedral angle, we take the arccosine of the dot product of this normal vector with the vector $(0, 0, 1)$, which is the unit normal vector of the xy-plane. Finally, we take the maximum over all dihedral angles. The input model states that this quantity should be bounded from above by a constant β which is strictly smaller than $\frac{\pi}{2}$.

Results

In Figure 6.5, the graphs with the results of our experiments are displayed. The maximum dihedral angles for the full terrains lie in the range $[31.815, 80.067]$ with an average of 62.304.

Conclusion

The graphs are very clear; for our input set, the dihedral angle does not become arbitrarily close to $\frac{\pi}{2}$ when n becomes very large, i.e., the triangles of the TIN will not become arbitrarily steep. In particular, the maximum value of the dihedral angle depends on the maximum height difference of adjacent DEM grid points. It could be larger for smaller n, since the global maximum and the global minimum of the DEM could be incident vertices in a very coarse TIN (they probably even will when the greedy vertex insertion method is used). But even then, in our terrains, there is a natural upper bound on the value of the dihedral angle. Therefore, we conclude that also the fourth assumption of the input model is satisfied by our terrains, and we call it realistic for this input set.

sectionComplexity of the visibility map

Besides verifying whether the input model is realistic, we also check whether the results that are obtained with this model in Chapter 5 conform with reality. In particular, we determine what the typical complexity is of the visibility map (VM) in terrains in practice.

The visibility map of a viewpoint on a terrain is the subdivision of that terrain into maximal connected components such that throughout each cell the viewpoint p is either visible or invisible. It has worst-case complexity $\Theta(n^2)$[131].

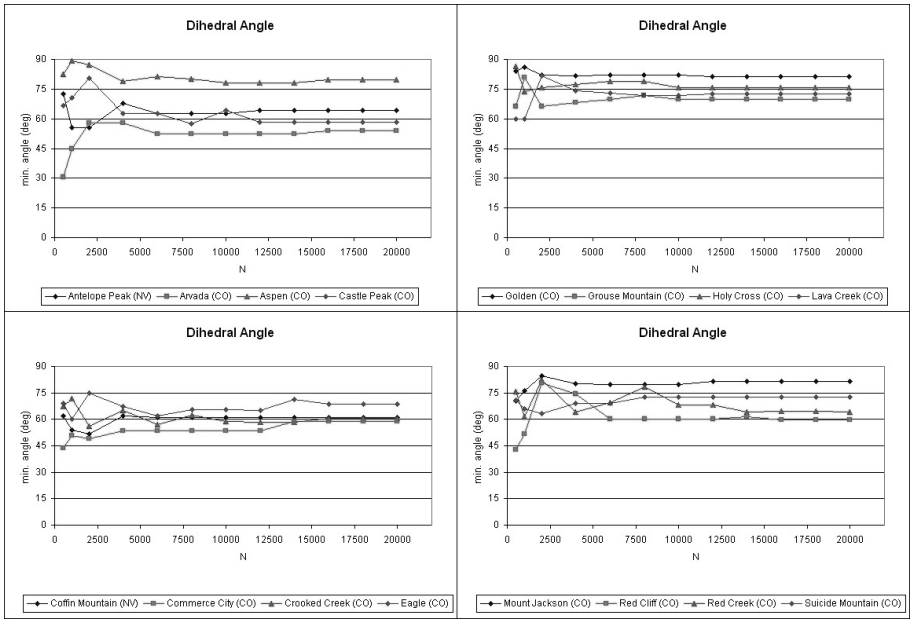


Figure 6.5: Dihedral angle as a function of n.

To analyze the complexity of the VM in practice, we compute the complexity of the transparent VM for our sixteen scenes, and for three different viewpoints. We pick these viewpoints such that they differ significantly in location: one is located very far away from the terrain, one close to the boundary of the terrain, and one right in the middle of the terrain. The first viewpoint should yield a VM similar to one of a viewpoint at infinity; the height at which the viewpoint is located is almost irrelevant in this case. We place the second viewpoint at the height that is halfway in the range of z-values of the vertices of the terrain. The third viewpoint is placed a little bit higher, at two-thirds in the range of z-values. We chose these three viewpoints in this way in an attempt to achieve that any other possible viewpoint will have a VM that resembles the VM of one of these three viewpoints. This is not the case for viewpoints that lie far above or below the terrain, but for such viewpoints, basically the entire terrain is visible, and thus the complexity is $O(n)$, which makes it unnecessary to study these cases any further.

Now, we first describe how we determine the complexity of the VM, and then we present the results that we obtained.

6.4.5 Computation

We compute the complexity of the VM by counting its vertices. A vertex of the visibility map of a point p directly corresponds to a line through p that is a common tangent of two terrain edges, see Chapter 5. When we assume that the triangles of the terrain do

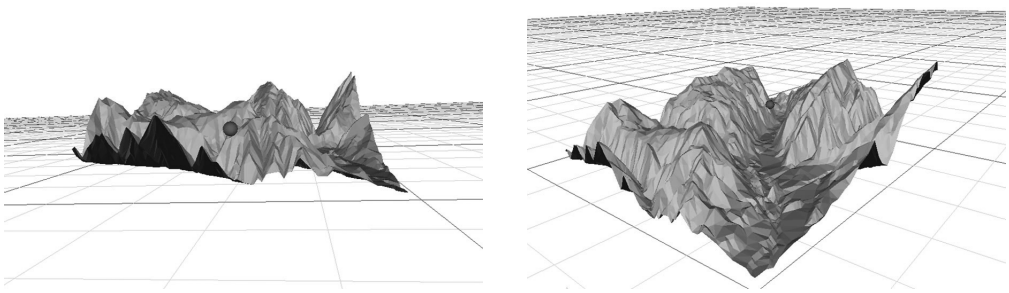


Figure 6.6: The second (left) and third (right) viewpoint for the scene Aspen (CO) with $n = 10000$.

not obstruct visibility, the VM becomes the *transparent visibility map*, which generally has higher complexity than the so-called *occluded visibility map*. We count the vertices of the transparent VM.

At the start of the algorithm, we initialize the VM complexity for the viewpoint p to be the number of vertices n_v, so that we don't have to count all incident edge pairs or coplanar edges. Our implementation iterates over all unique pairs of edges, and determines for each such pair whether or not they contribute a vertex, i.e., whether they have a common tangent. For two edges e_1 and e_2, we first construct the planes P_1 and P_2, that are defined by p and e_1 and p and e_2, respectively. The intersection of these two planes generally is a line l — if e_1 and e_2 are coplanar, P_1 and P_2 are equal. Note that P_1 and P_2 cannot be disjoint, since p is a common point. This line l does not necessarily intersect the two edges; it intersects both e_1 and e_2 if and only if they induce a transparent VM vertex.

Now we determine whether e_1 intersects P_1 and e_2 intersects P_2; if not, we finish for this edge pair and do not increase the complexity. If they do intersect, we switch to 2D and determine whether the 2D projections of e_1 and e_2 are consecutively intersected by the projection of l, i.e., whether they intersect l on the same side of p. Of course, they do not define a vertex of the VM of p if p has to view in different directions to see them. If this test also evaluates to true, then we increase the complexity by one and move on to the next edge pair. For a terrain with n triangles and $n_e = O(n)$ edges, we check $O(n^2)$ edge pairs, and thus the algorithm described above takes $O(n^2)$ time.

6.4.6 Results

In this section, we present the results of this vertex counting algorithm for the three different viewpoints. In Figure 6.7, the complexities of the VMs in all sixteens scenes, and for the three different viewpoints, are displayed as functions of n. They are all quite similar, and seem to grow just a little bit faster than linearly. Since the algorithm takes quadratic time and the full terrains have roughly 160,000 vertices, we have not included these terrains in the computations. Recall that in Chapter 5, it was shown

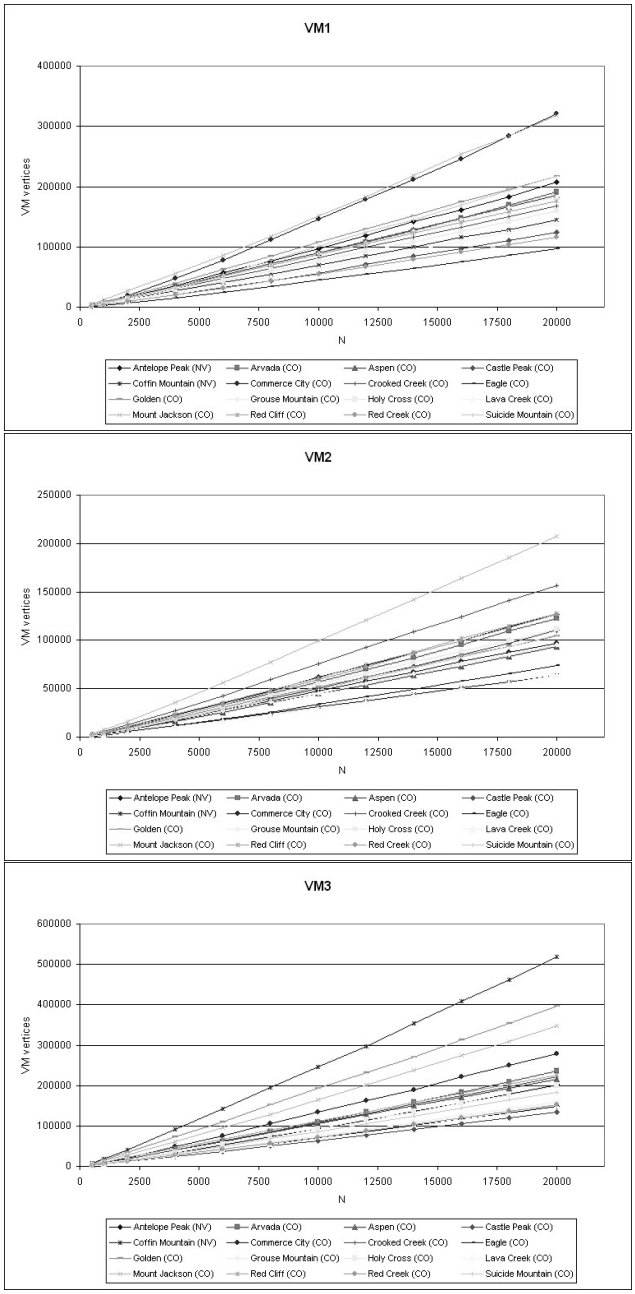


Figure 6.7: Complexities of the three visibility maps as functions of n.

that the complexity of the visibility map for a terrain that satisfies the input model has complexity $\Theta(n\sqrt{n})$ in the worst case.

To get a better idea of the growth rate of these complexities, we present graphs of the *approximated derivative* of the complexity for the first viewpoint. The other two viewpoints give similar results. This approximated derivative was computed discretely. Let c_{n_i} be the computed complexity for the number of triangles n_i. We approximate the derivate at this point with the central difference quotient:

$$\frac{1}{2}\frac{(c_{n_{i+1}} - c_{n_i})}{(n_{i+1} - n_i)} + \frac{1}{2}\frac{(c_{n_i} - c_{n_{i-1}})}{(n_i - n_{i-1})}.$$

We approximate the second derivative in the same way, from the values of the first derivative. The results are shown in Figures 6.8 and 6.9.

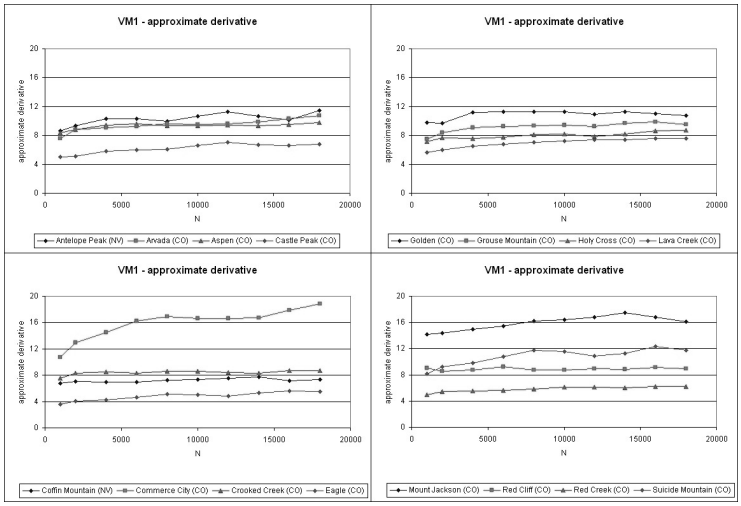


Figure 6.8: Approximated first derivative of the VM complexity (viewpoint 1) as a function of n.

All graphs in Figure 6.8 show an increasing growth rate, which means that in this case the VM complexity grows more than linearly with n. The growth rate for the scene Commerce City (CO) increases the fastest. Its graph of the approximated second derivative even starts growing again at the end. This scene has the following parameters: for n =16000, 18000, and 20000, it has fatness 0.213, edge length ratio 68 and dihedral angle 58.63. In other words, it satisfies the input model and we call it a realistic terrain. In this way, we obtain evidence that the visibility map in a realistic terrain can indeed have superlinear complexity, which was theoretically shown in Chapter 5. It has complexity $\Omega(n)$ by definition, and these graphs give the idea that it may be unlikely to find a visibility map with a complexity of $\omega(n\sqrt{n})$ in a realistic terrain.

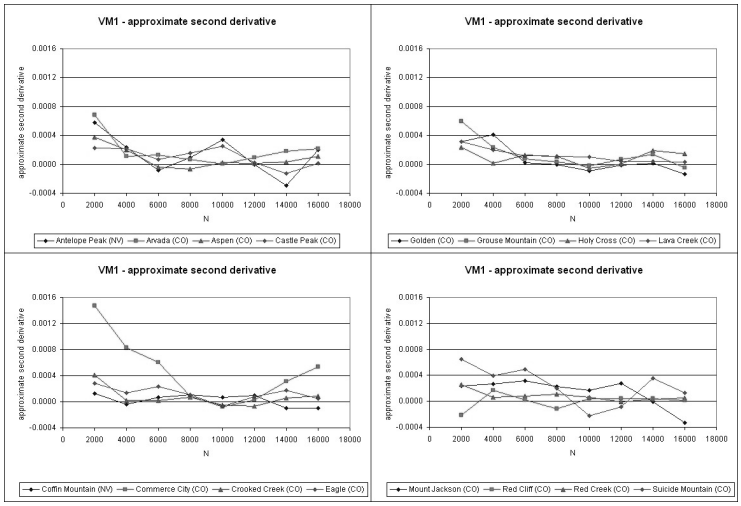


Figure 6.9: Approximated second derivative of the VM complexity (viewpoint 1) as a function of n.

6.5 Conclusions

We have studied 208 polyhedral terrains representing sixteen areas in the US. They are generated in such a way that they are similar to typical terrains that are used in GIS. We have observed that these particular terrains satisfy the input model of Chapter 5, in the sense that the parameters α, c, d, and β are all constants with respect to the number of triangles of the terrain, n.

The terrains that we used, and also the majority of the terrains that occur in applications, originate from a regular grid, that is, the xy-placements of their vertices are restricted. This implies that some input model parameters are naturally bounded from below resp. above. The main artifact is that the xy-projection of an edge of the terrain is never shorter than the DEM cell size. This immediately gives a lower bound on the fatness. The maximum length of a projected edge is the diagonal of the input domain, which implies that the edge length ratio is bounded from above by a constant (depending on N). The dihedral angle is also bounded from above, by a scene-specific value, as we discussed in Section 6.4.4.

Complexity bounds that are obtained with a realistic input model always depend on the value of the input model parameters. Therefore, it is not only desirable to have parameters that are constants with respect to n, but also to have relatively *small* parameters when they are used as upper bounds in input model assumptions and relatively *large* parameters when they are used as lower bounds. In our experiments, the parameter α is a lower bound. The smallest α that we encountered was 0.155 (Aspen (CO); $n = 1000$), . The parameter c is an upper bound, and the largest c that we found was 1.315 (all TINs for Golden (CO)). The parameter d is an upper bound, and

the largest d that we found was 467.7 (Commerce City (CO); $n = 500$)). Finally, the parameter β is also an upper bound, and the largest β in our experiments was 89.11 (Aspen (CO); $n = 1000$).

In Section 6.4.4, we counted vertices of the transparent visibility map. Its complexity is an upper bound on the complexity of the occluded visibility map. Of course, in GIS people will only be interested in the latter. Thus, it might be interesting to evaluate the complexity of the occluded visibility map in practice to see whether this VM also can attain superlinear complexity in realistic terrains.

We have studied terrains of various resolutions, that is, terrains whose vertices are all extracted from the same DEM, but that have different approximation errors. Another way to verify the input model would be to keep the approximation error the same, or to keep the resolution the same (e.g., one out of every 1000 grid points is a vertex) and to expand and shrink the DEM domain. We expect however, that these methods would give similar results to those of ours, since we implicitly also compare 16 terrains with equal resolution. We have created TINs for values of n in the range of 500 to 20,000. This is not as much as n will be in typical GIS applications, and of course the amount of data will keep growing in the future. We did not have a supercomputer to our disposal, and our computations quickly became more demanding with growing n. Our results do give some evidence that the input model from Chapter 5 is realistic for terrains from practice, but it still would be useful to obtain further such evidence by evaluating larger terrains.

Screenshots of the terrains

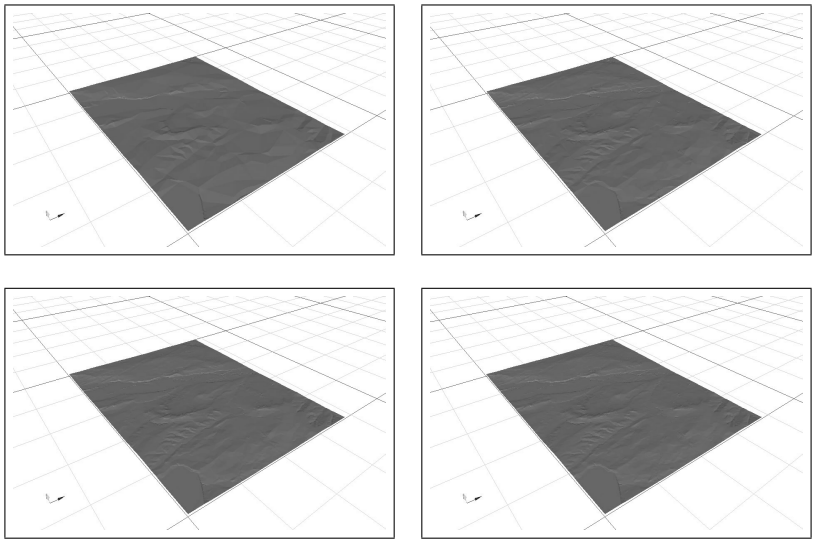


Figure 6.10: Arvada (CO).

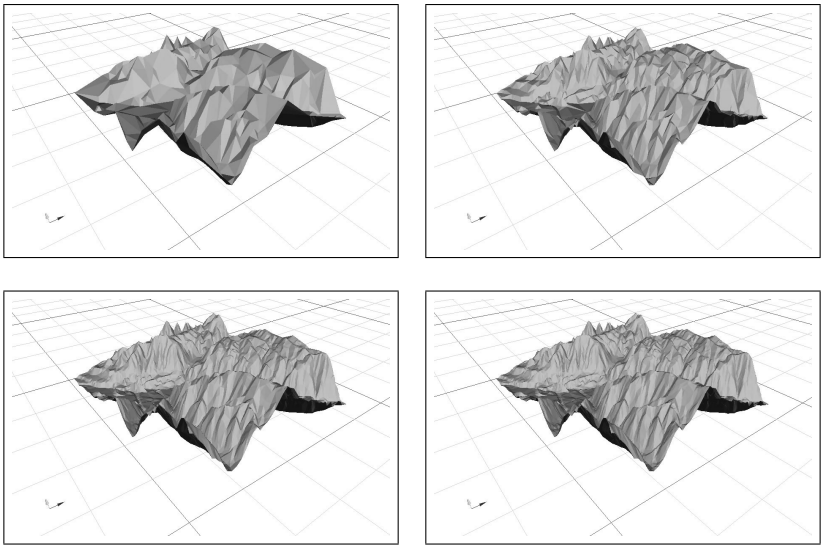


Figure 6.11: Aspen (CO).

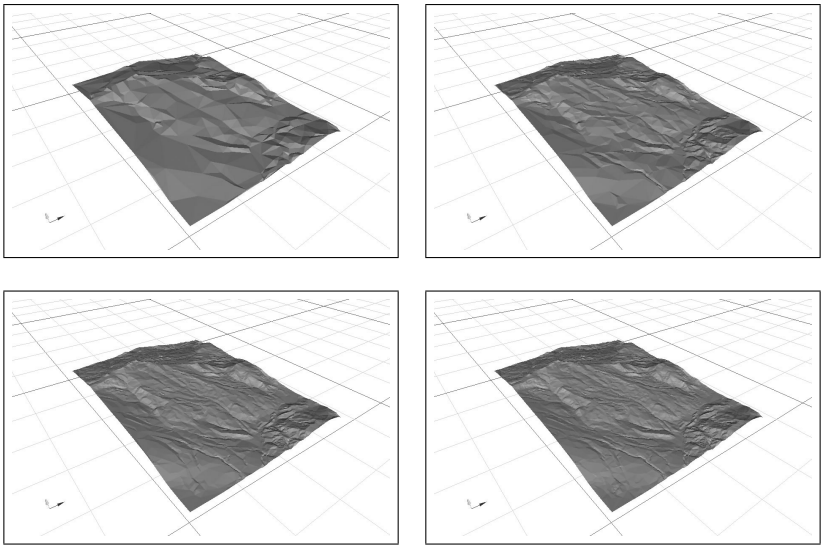


Figure 6.12: Castle Peak (CO).

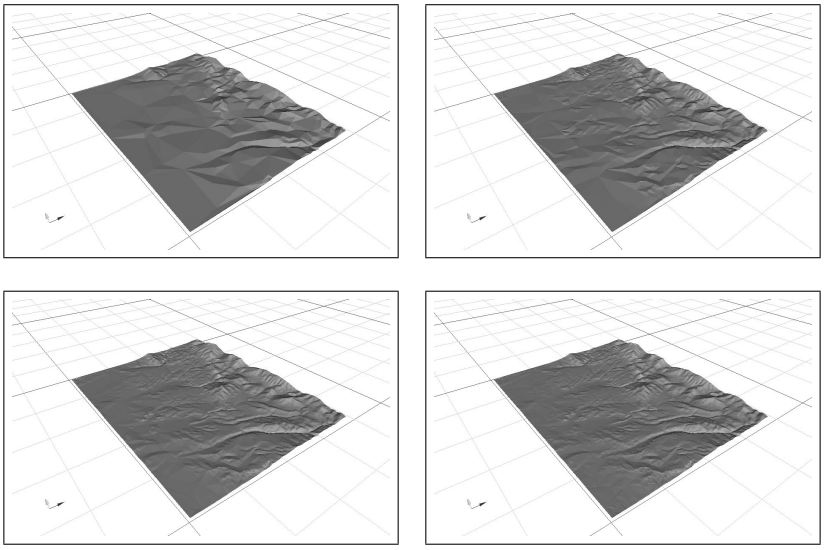


Figure 6.13: Coffin Mountain (NV).

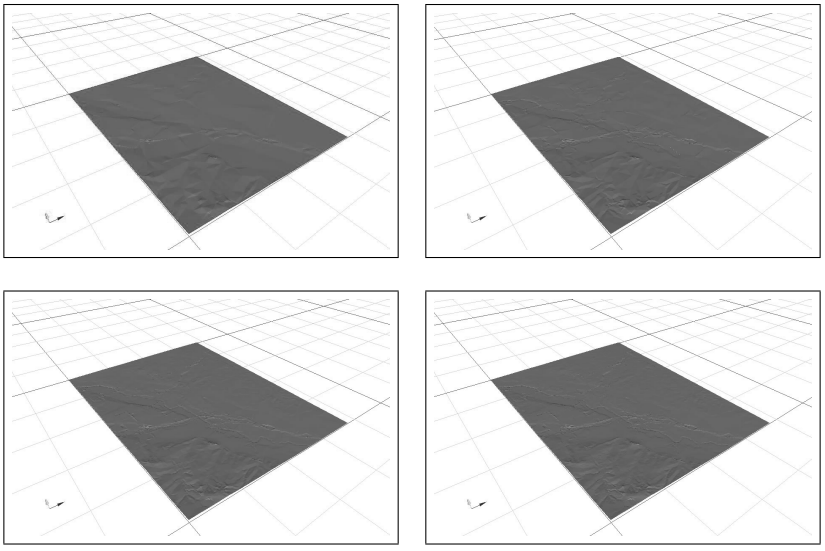


Figure 6.14: Commerce City (CO).

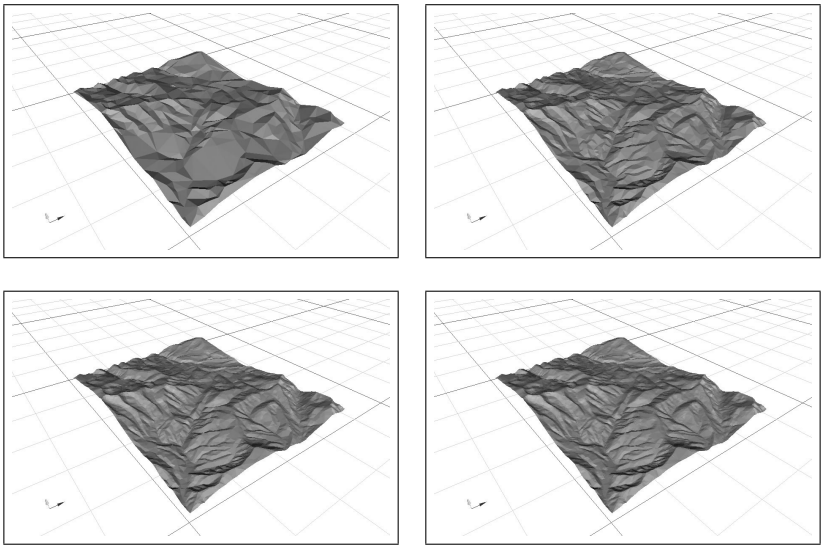


Figure 6.15: Crooked Creek (CO).

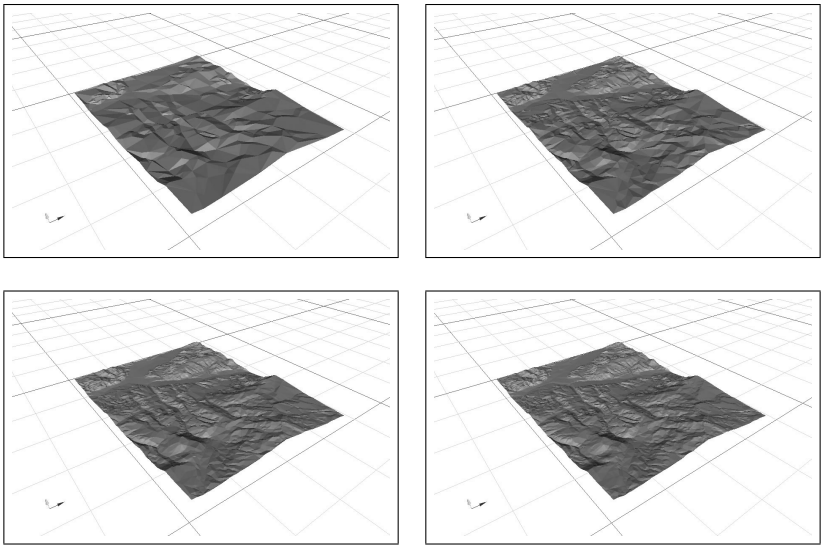


Figure 6.16: Eagle (CO).

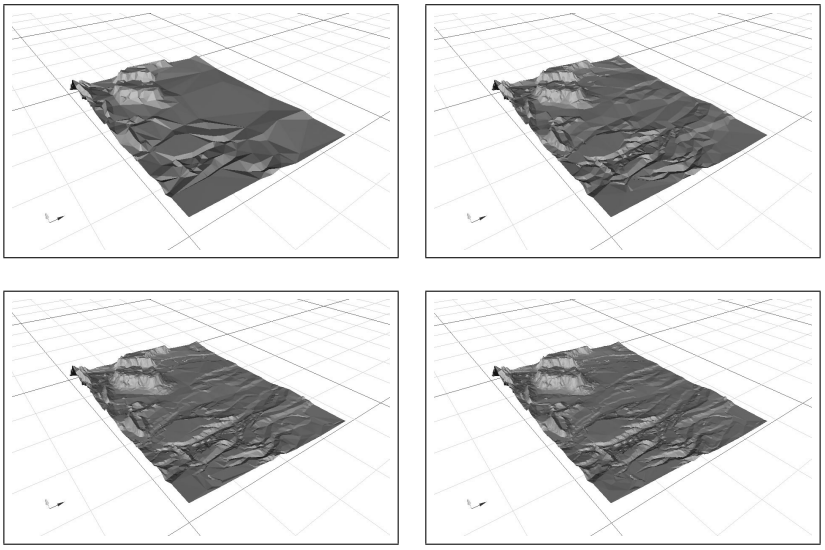


Figure 6.17: Golden (CO).

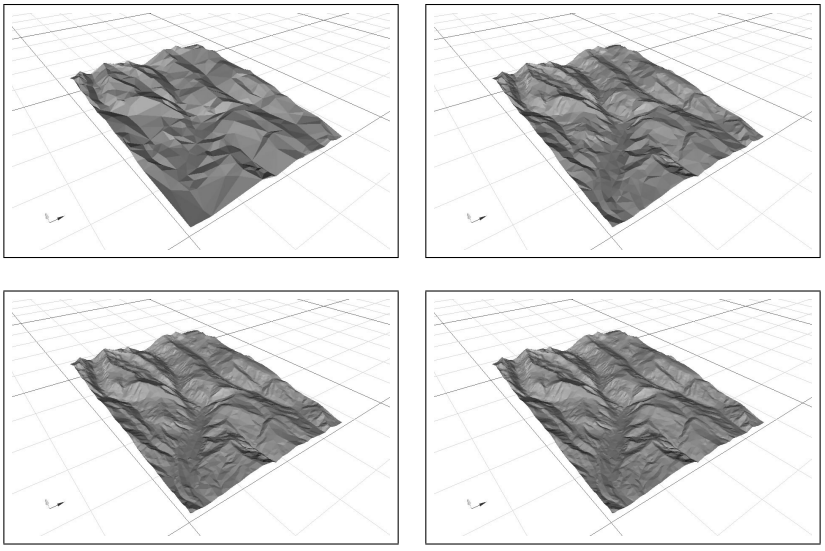


Figure 6.18: Grouse Mountain (CO).

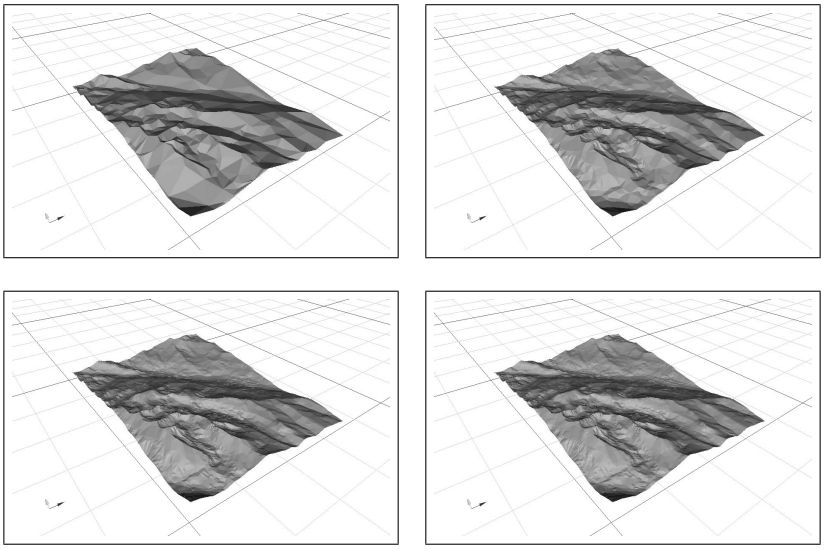


Figure 6.19: Mount of the Holy Cross (CO).

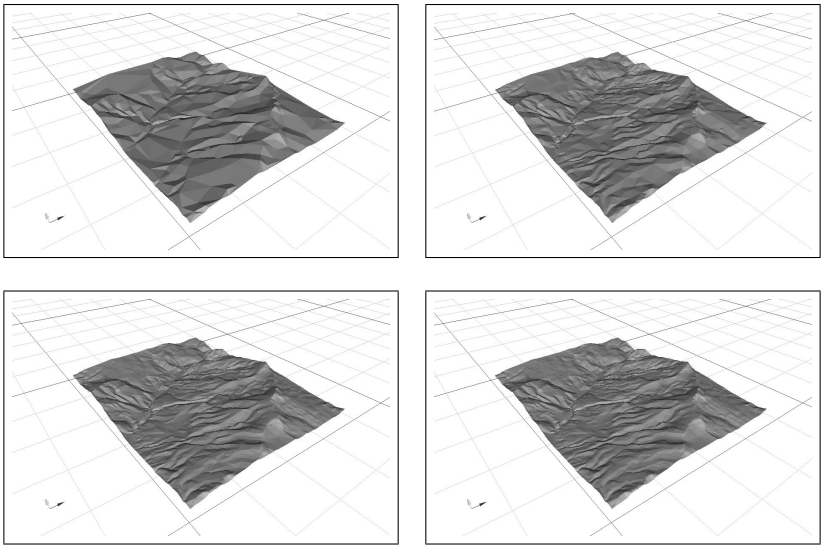


Figure 6.20: Lava Creek (CO).

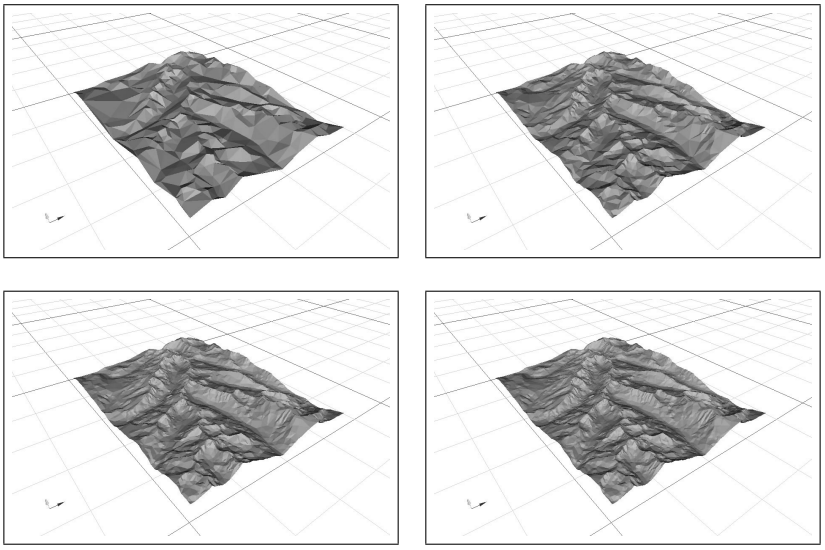


Figure 6.21: Mount Jackson (CO).

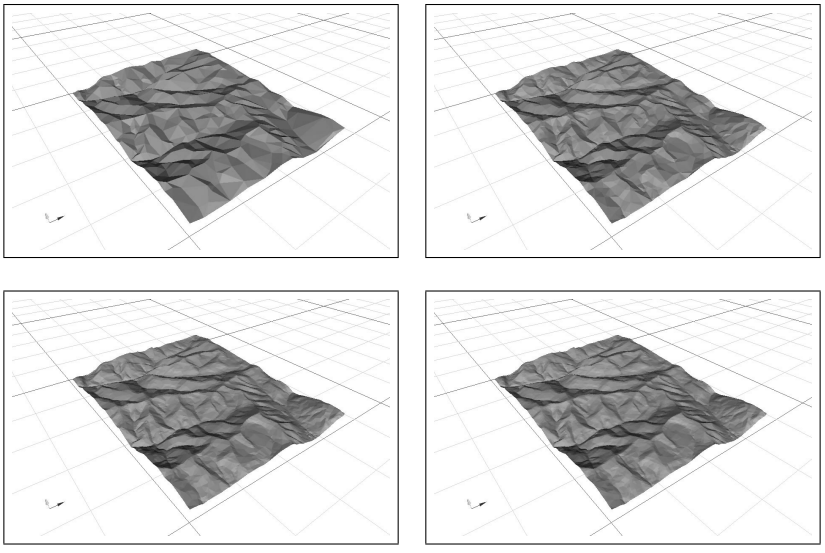


Figure 6.22: Red Cliff (CO).

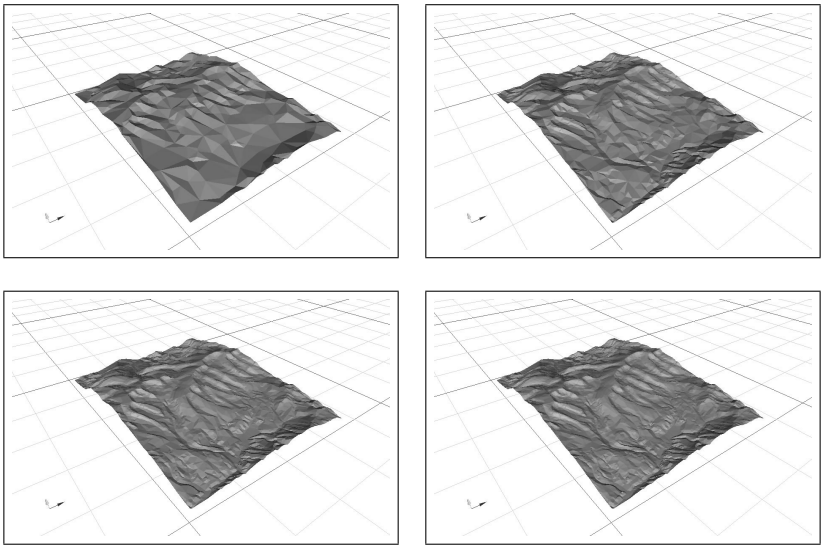


Figure 6.23: Red Creek (CO).

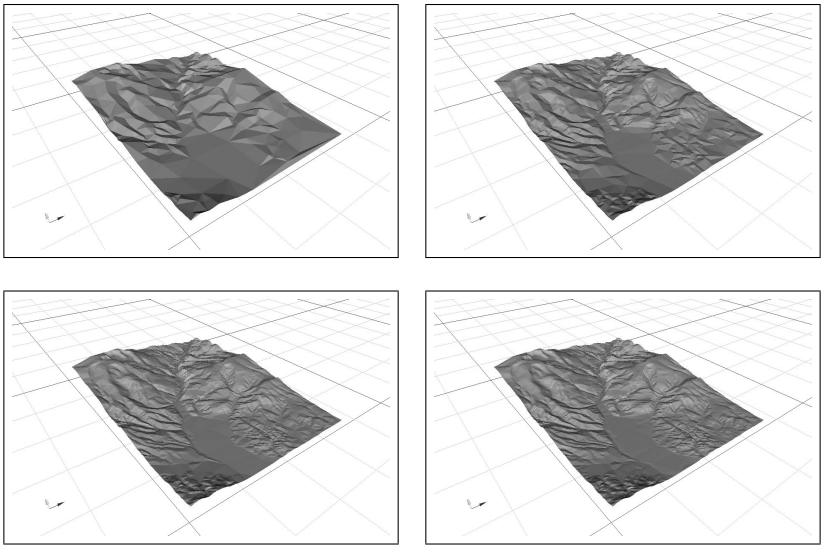


Figure 6.24: Suicide Mountain (CO).

Chapter 7

Conclusions

In this final chapter, we recapture the main results of this thesis, reflect on the contributions, including their limitations, and consider directions for future research on these topics. The first part of this chapter should be considered as an addition to the concluding sections Chapters 2 to 6, although there is some overlap in the treatment of the most important open problems.

Reflections on the contributions of this thesis

In Chapter 2, we showed that we can determine in $O(n^2 \log n)$ time whether a simple polygon admits a finite witness set, and if it does, then a minimum size witness set can be constructed in $O(n)$ time. We showed that these point witnesses have topologically similar locations for all finitely witnessable polygons. The main problem with this result is of course that not all simple polygons are finitely witnessable. Suppose for a moment that this were the case, then one of the main directions of future research would be to gain new insights to the art gallery problem using witness sets. Consider a simple polygon P, for which an $O(n)$ size witness set was given (or could efficiently be constructed). Then perhaps it would be possible to construct a minimum size guard set for P more efficiently, making use of (the location of) the witnesses. Maybe we could even find such a minimum guard set in polynomial time. Or otherwise, we could possibly develop better approximation algorithms, based on the existence of witness sets. But Chapter 2 showed that all such ideas will not work for all polygons, because finite witness sets need not exist. They would only work for the rather restricted class of finitely witnessable polygons.

We considered point witnesses in Chapter 2. We have also studied different types of witnesses to overcome the above mentioned problem of point witnesses. In particular, we studied witness sets consisting of line segments. Unfortunately for any definition of visibility that we tried out, we could find a polygon that did not admit a finite segment witness set. So it is doubtful that witness sets will lead to improved generic results with respect to the art gallery problem in the future.

In Chapter 3, we developed three efficient algorithms to determine different types

of mutual visibility of two connected sets of triangles on a polyhedral terrain. We studied the computation of complete, semi-complete, and partial intervisibility. We can imagine that intervisibility between sets of objects can be of importance in a number of applications; here, we mainly think of partial intervisibility. As we have already mentioned in Chapter 3, partial visibility of a path from a watchtower may be an interesting problem, but other variants may also be of interest, such as intervisibility of two paths on a terrain. We come back to the applicability of this algorithm later in this chapter. We studied intervisibility on (2.5-dimensional) polyhedral terrains, which was motivated by GIS applications. If we think of computer graphics as an application, then intervisibility amidst sets of triangles or more complex objects in 3D may also be interesting to consider. In that case, efficiency will have priority over exactness, and ease of implementation is essential.

In Chapter 4, we give upper and lower bounds on the combinatorial worst-case complexity of the strong and weak visibility maps of segments and triangles for sets of triangles in 3D. We also present almost optimal algorithms to compute all discussed structures. Among many other results, we show that the worst-case combinatorial complexity of the weak visibility map of a triangle in a general 3D scene is $\Theta(n^7)$. The auxiliary result for the complexity on a single triangle is similar to the earlier result shown by Wang and Zhu [193]. They show that the weak visibility region of a triangle on another triangle in a general 3D scene has complexity $\Theta(n^6)$ in the worst case. Their lower bound construction for a single triangle is substantially different in nature from our construction to obtain an $\Omega(n^7)$ lower bound for the whole scene. In particular, it is not clear how the construction by Wang and Zhu could be extended to a scene with n triangles. Moreover, our algorithms are significantly faster than those of Wang and Zhu. Wang and Zhu also obtain an $O(n^4)$ worst-case complexity bound for a single triangle when the scene is a polyhedral terrain. We prove the same bound in a slightly different way, and show that it is tight. The biggest open problem that remains after this chapter, is the linear size gap for the worst-case complexity of the weak visibility map in a polyhedral terrain. This problem turned out to be hard to solve, and extensive research unfortunately did not yield improved bounds.

In Chapter 5, we considered visibility and distance structures on polyhedral terrains, and tried to capture the reasons for the discrepancy between the theoretical time and complexity bounds and the ones we observe in practice. To this end, we introduced a realistic input model for terrains, and showed improved upper and lower bounds for these structures. This direction of research is very promising and attention is growing. For example, very recently studies have been presented on visibility maps of fat objects [17], and on shortest paths on 'realistic' polyhedra [170]. We elaborate more on realistic input models and their role in future visibility research in the next section.

In Chapter 6, we evaluated the parameters of the input model of Chapter 5. Moreover, we computed the visibility map for a set of terrains generated from GIS data. We have already mentioned the main open problems that remain after this study in the conclusions of that chapter, namely that the TINs that we evaluated are probably not large enough, and thus larger TINs should be evaluated to obtain more evidence to support our initial findings that our model is in fact realistic.

General reflections on (future) visibility research

As a start of our presentation of the general outlook on visibility research, we consider the art gallery problem. For over twenty years, there has been both extensive and intensive studies of (many variants of) this problem. We believe that research on the art gallery problem is close to finished. In the introduction, we mentioned many references to previous work on this problem; finding a minimum size guard set for a given polygon is NP-hard, and some approximation factors are not attainable in polynomial time, unless $\mathbb{P}$ equals $\mathbb{NP}$. On the other hand, there exist several approximation algorithms with approximation factor $O(\log c)$, where c is the size of the minimum guard set, and thus there is not much to be gained here anymore. We believe there are many other problems that are interesting and challenging to (combinatorial) visibility researchers, originating in the previously mentioned areas of graphics, GIS, and robotics.

One example from robotics is a problem that asks from a robot to localize itself within a known polygonal or polyhedral map of the environment, based on a set of the edges or faces that the robot has seen. To solve such a problem, the environment map should be processed based on (combinatorial) visibility information. In GIS, both urban and other areas are more and more digitally stored to facilitate computations that support planning decisions. Based on 3D city models, for example, horizon pollution can be analyzed in advance, that is, before a certain skyscraper or power plant is built. The rapidly emerging game industry deals with large virtual environments, and with (groups of) entities moving in those environments while performing a variety of tasks. We believe this field could benefit greatly from (research on) faster and more powerful computations concerning visibility.

The remainder of the second part of this chapter, in which we give our thoughts on the future of visibility research in computational geometry, has a common theme: *theory versus practice*.

Theoretical abstraction

As we mentioned at the very beginning of this thesis, theoretical computer science makes it necessary to abstract from practice to get to the mathematical foundations. The mathematical modeling of a problem may introduce undesirable effects, which should be evaluated and compared with reality. As an example, we wish to mention the issue of omnivisibility, i.e., the fact that in computational geometry, usually observers have the ability to look everywhere at the same time. This is of course no realistic assumption, since optical sensors in practice, e.g., in robotics, do not have this ability. The solutions to problems that adopt this assumption will therefore by definition not be used in practice. We believe that by getting feedback from the people working in the application areas that are the motivation behind theoretical visibility research, it is possible to obtain more realistic modeling concepts, and thus obtain theoretical results that better correspond with reality. For example, besides modeling the limitations of viewing sensors in terms of range and angle, one could introduce temporal characteristics to model a sensor that looks in different directions at different times.

A purely theoretical issue is that of extending visibility to dimensions higher than

three. In principle, the vast majority of visibility definitions in 2D or 3D, can be generalized to higher dimensional spaces. Of course, the intuitive feeling of what 'being seen', mutual visibility, and similar concepts mean will be lost, but still these definitions can be both valid and useful. It is imaginable that several problems from completely different application areas such as databases can be translated to visibility problems in d-D space. This idea is in the spirit of, e.g., orthogonal range searching that corresponds to a database query.

Implementation and practicality

The algorithm which we present in Chapter 2 to determine finite witnessability, is straightforward and easy to implement. The only question is of course whether anyone would ever consider it worthwhile to actually implement this algorithm, because of the drawbacks of the concept of witness sets which we mentioned above.

The algorithm for complete intervisibility from Chapter 3 uses complex data structures (multi-level cutting and partition trees, with associated structures) and will therefore be difficult to implement. The other two algorithms involve the computation of shadow maps of line segments, which is one of the subjects of Chapter 4. The algorithm for semi-complete intervisibility is also hard to implement, since it involves the computation of an arrangement of a set of algebraic curves on a triangle. We believe that the third and final problem that we consider in Chapter 3, namely partial intervisibility, is the one that is most likely to occur in practical applications. The good news is that our algorithm for this problem is by far the easiest to implement. Moreover, the algorithm is basically a repetition of determining intervisibility between two line segments, and is therefore not only conceptually very simple, but also very generally applicable. It can be used for many other types of intervisibility, such as visibility between two polygonal paths in 3-space. In fact, an algorithm which essentially does the same as ours was used to compute an approximate multi-visibility map for a set of line segments [46].

The algorithm for the strong visibility map of Chapter 4 is quite straightforward and consists of simple steps, and thus should not be hard to implement. The algorithms for the weak visibility map however, involve computations with reguli and arrangements of algebraic curves. In the CGAL community [37], there has been, and still is, much research activity on robust and exact geometric computing. This is an important direction for future research in computational geometry in its own right, but we also believe that the advances achieved there are of particular use to computations regarding visibility.

A final remark on practical issues that we wish to make is that of the use of the Graphical Processing Unit (GPU) in visibility computations. These days, the GPU is being used more and more for (parallel) computations other than visualization, especially in GIS (see e.g., [45]). Many visibility algorithms naturally deal with a large number of simple 3D geometric objects, which could possibly be processed in parallel. Since they also often have applications in graphics, we believe they are very well fit for adaption to computation by means of the GPU.

Discrepancy between theory and practice

We believe one the main issues in computational geometry research in the years to come should be to compare theory and practice, and get practical feedback on the obtained theoretical results. Note that we do not only refer to visibility research here. As we mentioned several times before in this thesis, there often is a large gap between time and complexity bounds that are observed in practice and those that are derived theoretically.

There are several ways to investigate and analyze this discrepancy. One of them is smoothed analysis [179, 180], which we already discussed in the conclusions of Chapter 5. We now discuss the two most important approaches to analyze and overcome this discrepancy in the last part of this chapter, while mentioning that we applied both of these techniques in this thesis.

The first method is to try to adopt a restricted theoretical model of reality, and use this to prove bounds that correspond better with practice. These models are called realistic input models, which have been mentioned at several places in this thesis. We developed a realistic input model for polyhedral terrains in Chapter 5 and used it to explain the theory-practice discrepancy for the visibility map and two distance structures on these terrains. In particular, we showed a tight $\Theta(n\sqrt{n})$ bound for the visibility map for a viewpoint in a polyhedral terrain, which approaches reality much better than the original $\Theta(n^2)$ bound [131]. This approach could definitely be rewarding for other topics in this thesis as well. As the most important example, we mention the results in Chapter 4, and in particular the $\Theta(n^7)$ worst-case complexity bound that we showed for the weak visibility map of a triangle in a general 3D scene. The lower bound construction for this result is a prime example of the reason that realistic input models were introduced: it describes a far-fetched and specific hypothetical situation which is very unlikely to occur in practice. It is largely dependent on a specific, regular distribution of the environment objects, as well as their long and skinny shape. The assumptions in a realistic input model prohibit such constructions and shapes and therefore eliminate these worst-case scenarios from the set of possible inputs. We also point out that one can adopt assumptions on the location and the shape of the view element as well, as opposed to merely the environment objects. There has been some recent attention, such as the earlier mentioned study on visibility maps of fat objects [17], but we believe, also based on the discouraging worst-case complexity bounds of Chapter 4, that there is still a lot to be gained from realistic input models in visibility research.

The second approach to relate theoretical and practical results is that of experimental verification. We performed this technique in Chapter 6, in which we evaluated the complexity of the visibility map in terrains that were generated from GIS data. The idea of empirically determining typical running times of algorithms and complexities of data structures is not new, but we believe it has the potential to be of much more use to computational geometers than it has been until now. It often is a challenge to find that construction that achieves the worst possible bound, but it could (or should) also be a challenge to develop mathematical models that are best satisfied by inputs from practice, and that also achieve the most realistic time and complexity bounds.

Of course, the ideal is to combine the two approaches and start a feedback interaction

process between them. The combined effect of theory corresponding with practice would then be achieved by both empirically evaluating the realistic input models to adopt them to better fit reality, and by checking whether the bounds that are obtained with the models correspond to the typical bounds that are observed empirically.

Acknowledgements

I thank the members of the reading committee for taking the time to read this thesis: Hazel Everett, Rolf Klein, Gert Vegter, Mark de Berg, and Jan van Leeuwen. Of course, I would also like to thank my promotor Mark Overmars and co-promotor Marc van Kreveld for their guidance and advice.

Bibliography

[1] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Approximate motion planning and the complexity of the boundary of the union of simple geometric figures. *Algorithmica*, 8:391–406, 1992.

[2] P.K. Agarwal, S. Bereg, O. Daescu, H. Kaplan, S. Ntafos, and B. Zhu. Guarding a terrain by two watchtowers. In *Proc. ACM Symp. Comput. Geom.*, pp. 346–355, 2005.

[3] P. K. Agarwal and M. Sharir. On the number of views of polyhedral terrains. *Discrete Comput. Geom.*, 12:177–182, 1994.

[4] J. Amanatides. Realism in computer graphics: A survey. *IEEE Comput. Graph. Appl.*, 7(1):44–56, 1987.

[5] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Minimum-link watchman tours. *Inform. Proc. Lett.*, 86(4):203–207, 2003.

[6] B. Aronov, A. Efrat, V. Koltun, and M. Sharir. On the union of κ-round objects in three and four dimensions. In *Proc. ACM Sympos. Comput. Geom.*, pp. 383–390, 2004.

[7] B. Aronov and J. O'Rourke. Nonoverlap of the star unfolding. *Discrete Comput. Geom.*, 8:219–250, 1992.

[8] M. Asada. Map building for a mobile robot from sensory data. *IEEE T. Syst. Man Cyb.* 20(6):1326–1336, 1990.

[9] T. Asano, T. Asano, L. J. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[10] T. Asano, S. K. Ghosh, and T. Shermer. Visibility in the plane. Ch. 19 in *Handbook of Computational Geometry*, J.-R Sack and J. Urrutia (Eds.), Elsevier Science Publishers B.W., pp. 829–876, 2000.

[11] B. Ben-Moshe, P. Carmi, and M.J. Katz. Computing all large sums-of-pairs in $\mathbb{R}^n$ and the discrete planar two-watchtower problem. *Inform. Process. Lett.*, 89:137–139, 2004.

[12] B. Ben-Moshe, M.J. Katz, J.S.B. Mitchell, and Y. Nir. Visibility preserving terrain simplification—an experimental study. *Comput. Geom. Theory Appl.*, 28:175–190, 2004.

[13] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28(9):643–647, 1979.

[14] M. de Berg. Linear size binary space partitions for uncluttered scenes. *Algorithmica*, 28:353–366, 2000.

[15] M. de Berg. *Ray Shooting, Depth Orders and Hidden Surface Removal.* LNCS 703, Springer-Verlag, 1993.

[16] M. de Berg. Vertical ray shooting for fat objects. In *Proc. ACM Symp. Comput. Geom.*, pp. 288–295, 2005.

[17] M. de Berg and C. Gray. Computing the visibility map of fat objects. *Algo. Data Struc.*, LNCS 4619, pp. 251–262, 2007.

[18] M. de Berg and C. Gray. Vertical ray shooting and computing depth orders for fat objects. In *ACM-SIAM Symp. Discrete Algo.*, pp. 494–503, 2006.

[19] M. de Berg, D. Halperin, M.H. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 12:30–53, 1994.

[20] M. de Berg, D. Halperin, M. Overmars, and M. van Kreveld. Sparse arrangements and the number of views of polyhedral scenes. *Internat. J. Comput. Geom. Appl.*, 7:175–195, 1997.

[21] M. de Berg, M. J. Katz, M. Overmars, A. F. van der Stappen, and J. Vleugels. Models and motion planning. *Comput. Geom. Theory Appl.*, 23:53–68, 2002.

[22] M. de Berg, M. J. Katz, A. F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. *Algorithmica*, 34:81–97, 2002.

[23] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, Germany, 2nd ed., 2000.

[24] M. T. de Berg and M. H. Overmars. Hidden surface removal for axis-parallel polyhedra. In *Proc. IEEE Symp. Found. Comput. Sci.*, pp.252–261, 1990.

[25] M. Bern. Hidden surface removal for rectangles. *J. Comp. Syst. Sci.*, 40:49–69, 1990.

[26] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point of view. *Algorithmica*, 11:360–378, 1994.

[27] S. Bespamyatnikh, Z. Chen, K. Wang, and B. Zhu. On the planar two-watchtower problem. In *Proc. Internat. Conf. Comput. Combin.*, pp. 121–130, 2001.

[28] B. K. Bhattacharya, G. Das, A. Mukhopadhyay, and G. Narasimhan. Optimally computing a shortest weakly visible line segment inside a simple polygon. *Comput. Geom. Theory Appl.*, 23(1):1–29, 2002.

[29] B. K. Bhattacharya and A. Mukhopadhyay. Computing in linear time a chord from which a simple polygon is weakly internally visible. *Algo. Comput.*, LNCS 1004, pp. 22–31, 1995.

[30] B. K. Bhattacharya and S. K. Ghosh. Characterizing LR-visibility polygons and related problems. *Comput. Geom. Theory Appl.*, 18(1):19–36, 2001.

[31] J. Bittner. *Hierarchical Techniques for Visibility Computations.* Ph.D. Thesis, Czech Technical University Prague, Oct. 2002.

[32] J. Bittner and P. Wonka. Visibility in Computer Graphics. *Env. Plan. B: Plan. Des.*, 30(5):729–756, 2003.

[33] J. Borenstein, B. Everett, and L. Feng. *Navigating mobile robots: Systems and techniques.* A. K. Peters, Ltd., Wellesley, MA, 1996.

[34] P. Bose, D. Kirkpatrick, Z. Li. Efficient algorithms for guarding or illuminating the surface of a polyhedral terrain. In *Proc. Can. Conf. Comput. Geom.*, pp. 217–222, 1996.

[35] P. Bose, T. Shermer, G. Toussaint, and B. Zhu. Guarding polyhedral terrains. *Comput. Geom. Theory Appl.*, 7(3):173–185, 1997.

[36] H. Brönnimann, H. Everett, S. Lazard, F. Sottile, and S. Whitesides. Transversals to line segments in three-dimensional space. *Discrete Comput. Geom.*, 34(3):381–390, 2005.

[37] Computational Geometry Algorithms Library. `http://www.cgal.org`.

[38] B. Chazelle, H. Edelsbrunner, L.J. Guibas, M. Sharir, and J. Stolfi. Lines in space: Combinatorics and algorithms. *Algorithmica*, 15:428–447, 1996.

[39] Z. Chen and J.A. Guevara. System selection of very important points (VIP) from digital terrain models for constructing triangular irregular networks. In *Proc. Internat. Sympos. Comput.-Assist. Cartog.*, pp. 50–56, 1988.

[40] J. Chen and Y. Han. Shortest paths on a polyhedron. *Internat. J. Comput. Geom. Appl.*, 6(2):127–144, 1996.

[41] O. Cheong and R. van Oostrum. The visibility region of points in a simple polygon. In *Canad. Comf. Comp. Geom.*, pp. 87–90, 1999.

[42] N. Chin and S. Feiner. Fast object-precision shadow generation for areal light sources using BSP trees. In *Proc. Sympos. Inter. 3D Graph.*, pp. 21–30, 1992.

[43] V. Chvátal. A combinatorial theorem in plane geometry. *J. Combin. Theory Ser. B*, 18:39–41, 1975.

[44] R. Cole and M. Sharir, Visibility problems for polyhedral terrains, *J. Symbolic Comput.*, 7: 11–30, 1989.

[45] N. Coll, M. Fort, N. Madern, J.A. Sellarès. Computing terrain multi-visibility maps for a set of view segments using graphics hardware. In *Proc. Int. Conf. Comput. Sci. Appl.* 1:81–90, 2006.

[46] N. Coll, M. Fort, N. Madern and J.A. Sellarès. Multi-visibility maps of triangulated terrains. *Internat. J. Geogr. Inform. Sci.*, 21(10):1115-–1134, 2007.

[47] M. C. Couto, C. C. de Souza, and P. J. de Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *Proc. Braz. Symp. Comput. Graph. Image Proc.*, pp. 87–94, 2007.

[48] I.J. Cox and G.T. Wilfong, eds. *Autonomous Robot Vehicles.* Springer Verlag, 1990.

[49] K.-Y. Chwa, B.-C. Jo, C. Knauer, E. Moet, R. van Oostrum, and C.-S. Shin. Guarding art galleries by guarding witnesses. *Internat. J. Comput. Geom. Appl.*, 16(2/3):205–226, 2006.

[50] G. Das, P .J. Heffernan, and G. Narasimhan. LR-visibility in polygons. *Comput. Geom. Theory Appl.*, 7(1):37–57, 1997.

[51] A. Deshpande, T. Kim, E. D. Demaine, and S. E. Sarma. A pseudopolynomial time $O(\log n)$-approximation algorithm for art gallery problems. In Alg. Data Struc., LNCS 4619, pp. 163–174, 2007.

[52] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(2):237–267, 2002.

[53] F. Dé vai. Quadratic bounds for hidden line elimination. In *Proc. ACM Symp. Comput. Geom.*, 269–275, 1986.

[54] D.P. Dobkin and D.G. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. In *Proc. Internat. Colloq. Automata Lang. Program.*, LNCS 443, pp. 400–413, 1990.

[55] G. Drettakis and E. Fiume. A fast shadow algorithm for area light sources using backprojection. *Comput. Graph.*, 28:223–230, 1994.

[56] F. Duguet and G. Drettakis. Robust epsilon visibility. In *Proc. ACM SIGGRAPH*, pp. 567–575, 2002.

[57] F. Durand. *3D Visibility: analytical study and applications.* Ph.D. Thesis, Université Joseph Fourier, Grenoble I, Jul. 1999.

[58] F. Durand, G. Drettakis, and C. Puech. Fast and accurate hierarchical radiosity using global visibility. *ACM T. Graph.*, 18(2):128–170, 1999.

[59] F. Durand, G. Drettakis, and C. Puech. The 3D visibility complex. *ACM T. Graph.*, 21(2):176–206, 2002.

[60] F. Durand, G. Drettakis, and C. Puech. The visibility skeleton: a powerful and efficient multi-purpose global visibility tool. In *Int. Conf. Comput. Graph. Interact. Tech.*, pp. 89–100, 1997.

[61] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990.

[62] A. Efrat. The complexity of the union of (α, β)-covered objects. *SIAM J. Comput.*, 34:775–787, 2005.

[63] A. Efrat and S. Har-Peled. Guarding galleries and terrains. Inform. Proc. Lett., 100(6):238-–245, 2006.

[64] D. Eggert, K. Bowyer, and C. Dyer. Aspect graphs: State-of-the-art and applications in digital photogrammetry. In *Proc. Congr. Internat. Soc. Photogramm. Remote Sens. B5*, pp. 633–645, 1992.

[65] S. Eidenbenz. *(In-)Approximability of visibility problems on polygons and terrains.* Ph.D. Thesis, Institute for Theoretical Computer Science, ETH Zürich, 2000.

[66] S. Eidenbenz, C. Stamm, P. Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica* 31:79-–113, 2001.

[67] J. Erickson. Local polyhedra and geometric graphs. *Comput. Geom. Theory Appl.*, 31:101–125, 2005.

[68] P.F. Fisher. Algorithm and implementation: Uncertainty in viewshed analysis, *Internat. J. Geogr. Inform. Syst.*, 7:331–347, 1993.

[69] P.F. Fisher. Reconsideration of the viewshed function in terrain modelling. *Geogr. Syst.* 3:33–58, 1996.

[70] P.F. Fisher. Stretching the viewshed. In *Proc. Int. Symp. Spat. Data Handl.*, pp. 725–738, 1994.

[71] P.-O. Fjällström. Polyhedral approximation of bivariate functions. In *Proc. Canad. Conf. Comput. Geom.*, pp. 187—190, 1991.

[72] P.-O. Fjällström. Evaluation of a Delaunay-based method for surface approximation. *Comput. Aided Des.*, 25(11):711–719, 1993.

[73] L. De Floriani, B. Falcidieno, C. Pienovi, D. Allen, and G. Nagy. A visibility-based model for terrain features. In *Proc. Internat. Sympos. Spat. Data Handl.*, pp. 235–250, 1986.

[74] L. De Floriani and P. Magillo. Intervisibility on terrains. In *Geo. Inform. Syst.: Princ. Tech. Man. Appl.*, P.A.Longley, M.F.Goodchild, D.J.Maguire, and D.W.Rhind (Eds.), John Wiley & Sons, pp. 543–556, 1999.

[75] L. De Floriani, G. Nagy, and E. Puppo. Computing a line-of-sight network on a terrain model. In *Proc. Int. Symp. Spat. Data Handl.*, pp. 672–681, 1992.

[76] R. J. Fowler and J. J. Little. Automatic extraction of irregular network digital terrain models. In *Proc. SIGGRAPH*, pp. 199–207, 1979.

[77] W.R. Franklin and C.K. Ray. Higher isn't necessarily better: Visibility algorithms and experiments. In *Proc. Int. Symp. Spat. Data Handl.*, pp. 751–763, 1994.

[78] M. L. Fredman and B. Weide. On the complexity of computing the measure of $\bigcup[a_i, b_i]$. *Commun. ACM*, 21:540–544, 1978.

[79] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *Proc. Conf. Comput. Graph. Interact. Tech.*, pp. 124–133, 1980.

[80] H. Fuchs, G. D. Abram, E. D. Grant. Near real-time shaded display of rigid objects. In *Proc. Conf. Comput. Graph. Interact. Tech.*, pp. 65–72, 1983.

[81] A. Gajentaan and M.H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5:165–185, 1995.

[82] J. Garcia-Lopez, P. A. Ramos. A Unified Approach to Conic Visibility. *Algorithmica*, 28(3):307–322, 2000.

[83] M. Garland and P. Heckbert. Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, Carnegie Mellon University, 1995.

[84] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *Proc. SIGGRAPH*, pp. 209–216, 1997.

[85] B. P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. *Int. J. Robot. Res.*, 25(4):299–315, 2006.

[86] L. Gewali, A. Meng, Joseph S. B. Mitchell, and S. Ntafos. Path planning in $0/1/\infty$ weighted regions with applications. *ORSA J. Comput.*, 2(3):253–272, 1990.

[87] S. Ghali, E. Fiume, and H.-P. Seidel. Object-space, connectivity-preserving, shadow computation. In *Proc. Conf. Vis. Mod. Visualiz.*, pp. 281–289, 2000.

[88] S. Ghosh. Approximation algorithms for art gallery problems. In *Proc. Can. Inform. Proc. Soc. Congr.*, 1987.

[89] S. K. Ghosh. *Visibility Algorithms in the Plane.* Cambridge University Press, 2007.

[90] S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs. In *Proc. IEEE Sympos. Found. Comput. Sci.*, pp. 11-–19, 1987.

[91] Z. Gigus, J. Canny, and R. Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE T. Patt. An. Machine Intell.*, 13(6):542–551, 1991.

[92] A. Glassner (Ed.). *An Introduction to Ray Tracing.* Academic Press, 1989.

[93] H. H. Gonzalez-Baños, L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, R. Motwani, and C. Tomasi Motion planning with visibility constraints: Building autonomous observers. *Int. Symp. Robot. Res.*, pp. 95–101, 1998.

[94] M. T. Goodrich. A polygonal approach to hidden-line and hidden-surface elimination. *CVGIP: Graph. Mod. Image Proc.*, 54:1–12, 1992.

[95] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. In *Works. Algo. Data Struc.* (LNCS 1272), pp. 17-–30, 1997.

[96] R. H. Güting and T. Ottmann. New algorithms for special cases of the hidden line elimination problem. *Comp. Vision, Graph. Image Proc.*, 40:188–204, 1987.

[97] E. Györi, F. Hoffmann, K. Kriegel, T. C. Shermer. Generalized guarding and partitioning for rectilinear polygons. *Comput. Geom.*, 6:21–44, 1996.

[98] D. Halperin. Arrangements. Ch. 24 in *Handbook of Discrete and Computational Geometry*, 2nd ed. J. E. Goodman and J. O'Rourke (Eds.), pp. 529-562, CRC Press, 2004.

[99] J.-M. Hasenfratz, M. Lapierre, N. Holzschuch, and F. Sillion. A survey of real-time soft shadows algorithms. *Comp. Graphics Forum*, 22(4):753–774, 2003.

[100] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical Report CMU-CS-95-194, Carnegie Mellon University, 1995.

[101] J. Heintz, T. Reico, and M.-F. Roy. Algorithms in real algebraic geometry and applications to computational geometry. *Discrete Comput. Geom. DIMACS Spec. Year*, pp. 137–163, 1991.

[102] M. Heller. Triangulation algorithms for adaptive terrain modeling. In *Proc. Internat. Symp. Spat. Data Handl.*, pp. 163-–174, 1990.

[103] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inform. Process. Lett.*, 33:169–174, 1989.

[104] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.

[105] F. Hoffmann and M. Kaufmann. On the rectilinear art gallery problem — algorithmic aspects. In *Proc. Work. Graph.*, pp. 239–250, 1990.

[106] H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH*, pp. 99—108, 1996.

[107] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[108] S. Kapoor. Efficient computation of geodesic shortest paths. In *Proc. ACM Sympos. Theory Comput.*, pp. 770–779, 1999.

[109] M.J. Katz. 3-D vertical ray shooting and 2-D point enclosure, range searching, and arc shooting amidst convex fat objects. *Comput. Geom. Theory Appl.*, 8:299–316, 1998.

[110] M. J. Katz, M. H. Overmars and M. Sharir. Efficient hidden surface removal for objects with small union size. *Comput. Geom.*, 2:223–234, 1992.

[111] J. J. Koenderink and A. J. van Doorn. The singularities of the visual mapping. *Bio. Cyber.*, 24(1):51—59, 1976.

[112] J. J. Koenderink and A. J. van Doorn. The internal representation of solid shape with respect to vision. *Biolog. Cybernet.*, 32:211–216, 1979.

[113] M. van Kreveld. Variations on sweep algorithms: efficient computation of extend ed viewsheds and classifications. In *Proc. Int. Symp. Spat. Data Handl.*, pp. 843–855, 1996.

[114] M. van Kreveld. On fat partitioning, fat covering, and the union size of polygons. *Comput. Geom. Theory Appl.*, 9(4):197–210, 1998.

[115] M. van Kreveld, E. Moet, and R. van Oostrum. Region inter-visibility in terrains. In *Abstr. Europ. W. Comput. Geom.*, pp. 155–158, 2004.

[116] B. J. A. Kröse and R. Bunschoten. Probabilistic localization by appearance models and active vision. In *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 2255–2260, 1999.

[117] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[118] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

[119] S. M. LaValle, B. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. *Int. J. Comput. Geom. Appl.*, 12(1-2):87-–113, 2002.

[120] D. T. Lee. Proximity and reachability in the plane. Report R-831, Dept. Elect. Engrg., Univ. Illinois, Urbana, USA, 1978.

[121] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE T. Inform. Theory*, 32:276–282, 1986.

[122] D. T. Lee and F. P. Preparata. An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26(3):415–421, 1979.

[123] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393–410, 1984.

[124] J. Lee. Analyses of visibility sites on topographic surfaces. *Internat. J. Geogr. Inform. Syst.*, 5:413–430, 1991.

[125] J. Lee. A drop heuristic conversion method for extracting irregular networks from digital elevation models. In *Proc. GIS/LIS*, pp. 30–39, 1989.

[126] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proc. IEEE Int. W. Intell. Rob. Syst.*, pp. 1442–1447, 1991.

[127] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox. Dynamic map building for an autonomous mobile robot. *Int. J. Robot. Res.*, 11(4):89-–96, 1992.

[128] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *Proc. IEEE Vis.*, pp. 279-–286, 1998.

[129] D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, 21(3):24–35, 2001.

[130] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. *SIAM J. Comput.*, 23:154–169, 1994.

[131] M. McKenna. Worst-case optimal hidden surface removal. *ACM T. Graph.*, 6:19–28, 1987.

[132] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.

[133] J. S. B. Mitchell, D. M. Mount, and S. Suri. Query-sensitive ray shooting. *Internat. J. Comput. Geom. Appl.*, 7(4):317–347, 1997.

[134] E. Moet, M. van Kreveld, and R. van Oostrum. Region intervisibility in terrains. *Internat. J. Comput. Geom.*, 17(4):331–347, 2007.

[135] E. Moet, C. Knauer, and M. van Kreveld. Visibility maps of segments and triangles in 3D. *Comput. Geom. Theory Appl.*, 39(3):163–177, 2008.

[136] E. Moet, C. Knauer, and M. van Kreveld. Visibility maps of segments and triangles in 3D. Technical Report UU-CS-2005-049, Universiteit Utrecht, 2005.

[137] E. Moet, M. van Kreveld, and A. F. van der Stappen. On realistic terrains. To appear in *Comput. Geom. Theory Appl.*, 2008/2009.

[138] E. Moet, M. van Kreveld, and A. F. van der Stappen. On realistic terrains. In *Proc. ACM Sympos. Comput. Geom.*, pp. 177–186, 2006.

[139] E. Moet. Experimental verification of a realistic input model for polyhedral terrains. Report UU-CS-2007-052, Universiteit Utrecht, 2007.

[140] D. Mount. Geometric intersection. Ch. 38 in *Handbook of Discrete and Computational Geometry*, 2nd ed. J. E. Goodman and J. O'Rourke (Eds.), pp. 857–876, CRC Press, 2004.

[141] K. Mulmuley. An efficient algorithm for hidden surface removal. *Comput. Graph.*, 23:379–388, 1989.

[142] K. Mulmuley. A fast planar partition algorithm, I. *J. Symbolic Comput.*, 10(3-4):253–280, 1990.

[143] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, 1993.

[144] R. Murrieta, B. Tovar, and S. Hutchinson. A sampling-based motion planning approach to maintain visibility of unpredictable targets. *Autonomous Robots*, 19(3):285–300, 2005.

[145] B. J. Nilsson. Guarding art galleries — methods for mobile guards. Ph.D. Thesis, Lund University, 1995.

[146] N. Nilsson. A mobile automaton: An application of artifical intelligence techniques. In *Proc. Int. Joint Conf. Art. Intell.*, pp. 509–520, 1969.

[147] S. Ntafos. Watchman routes under limited visibility. *Comput. Geom. Theory Appl.*, 1(3):149–170, 1992.

[148] O. Nurmi. A fast line sweep algorithm for hidden line elimination. *BIT*, 25(3):466–472, 1985.

[149] J. O'Rourke. *Art Gallery Theorems and Algorithms*. (Int. Ser. Monog. Comput. Sci.), Oxford University Press, 1987.

[150] J. O'Rourke. A note on fully-turned vertex pi-lights. TR 056, Smith College, 1997.

[151] J. O'Rourke. Visibility. Ch. 28 in *Handbook of Discrete and Computational Geometry*, 2nd ed. J. E. Goodman and J. O'Rourke (Eds.), pp. 643–664, CRC Press, 2004.

[152] M. H. Overmars and M. Sharir. Merging visibility maps. *Comput. Geom.: Theory Appl.*, 1(1):35–49, 1991.

[153] M. H. Overmars and A. F. van der Stappen. Range searching and point location among fat objects. *J. Algorithms*, 21:629–656, 1996.

[154] M. S. Paterson and F. F. Yao. Binary space partitions with applications to hidden surface removal and solid modeling. *Discrete Comput. Geom.* 5:485–503, 1990.

[155] M. Pellegrini. Ray shooting on triangles in 3-space. *Algorithmica*, 9:471–494, 1993.

[156] M. Pellegrini. Ray shooting and lines in space. Ch. 37 in *Handbook of Discrete and Computational Geometry*, 2nd ed. J. E. Goodman and J. O'Rourke (Eds.), pp. 839–856, CRC Press, 2004.

[157] S. PetitJean. Computing exact aspect graphs of smooth objects bounded by smooth algebraic surfaces. Report UIUC-BI-AI-RCV-92-04, Univ. Illinois, Urbana-Champaign, June 1992.

[158] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation.* Morgan Kaufmann, 2004.

[159] W. H. Plantinga and C. R. Dyer. An algorithm for constructing the aspect graph. In *IEEE Symp. Found. Comput. Sci.*, pp. 123-–131, 1986.

[160] H. Plantinga and C. R. Dyer. The asp: a continuous viewer-centered representation for 3D object recognition. In *Proc. Internat. Conf. Comput. Vis.*, pp. 626–630, 1987.

[161] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *Internat. J. Comput. Vis.* 5(2):137-–160, 1990

[162] M. Pocchiola and G. Vegter. The visibility complex. *Int. J. Comput. Geom. Appl.*, 6(3):279–308, 1996.

[163] M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudotriangulations. *Discrete Comput. Geom.*, 16(4):419–453, 1996.

[164] H. Pottmann and J. Wallner. *Computational Line Geometry.* Springer Verlag, Berlin, 2001.

[165] J. Reif and S. Sen. An efficient output-sensitive hidden surface removal algorithm and its parallelization. In *Proc. ACM Symp. Comput. Geom.*, pp. 193–200, 1988.

[166] Ruimtelijk Planbureau, Den Haag. *De zichtbaarheid van de Belle van Zuylentoren.* `http://www.rpb.nl/upload/documenten/belle_van_zuylen.pdf`

[167] S. Sachs, S. M. LaValle, and S. Rajko. Visibility-based pursuit-evasion in an unknown planar environment. *Int. J. Robot. Res.*, 23(1):3–26, 2004.

[168] J. R. Sack and S. Suri. An optimal algorithm for detecting weak visibility of a polygon. *T. Comput.*, 39(10):1213–1219, 1990.

[169] A. Schmitt. Time and space bounds for hidden line and hidden surface algorithms. *Eurograph.*, pp. 43–56, 1981.

[170] Y. Schreiber. Shortest paths on realistic polyhedra. In *Proc. ACM Symp. Comput. Geom.*, pp. 74–83, 2007.

[171] Y. Schreiber and M. Sharir. An optimal-time algorithm for shortest paths on a convex polytope in three dimensions. In *Proc. ACM Sympos. Comput. Geom.*, pp. 30–39, 2006.

[172] O. Schwarzkopf and J. Vleugels. Range searching in low-density environments. *Inform. Process. Lett.*, 60:121–127, 1996.

[173] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.

[174] M. Sharir and M. H. Overmars. A simple output-sensitive algorithm for hidden surface removal. *ACM T. Graph.*, 11:1–11, 1992.

[175] T. Shermer. Recent results in art galleries. In *Proc. IEEE*, 80(9):1384–1399, 1992.

[176] P. Shirley and K. R. Morley. *Realistic Ray Tracing,* 2nd ed. A.K. Peters, 2001.

[177] R. C. Smith and P. Cheeseman On the representation and estimation of spatial uncertainty. *Int. J. Robot. Res.* pp. 56–68, 1987.

[178] P. Sorensen and D. Lanter. Two algorithms for determining partial visibility and reducing data structure induced error in viewshed analysis. In *Phot. Eng. Remote Sens.* 28:1129–1132, 1993.

[179] D. A. Spielman and S.-H. Teng. Smoothed Analysis: Motivation and discrete models. In *Proc. W. Algo. Data Struc.*, pp. 256–270, 2003.

[180] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[181] A. F. van der Stappen. *Motion planning amidst fat obstacles.* Ph.D. Thesis, Universiteit Utrecht, 1994.

[182] A. F. van der Stappen, M. H. Overmars, M. de Berg, and J. Vleugels. Motion planning in environments with low obstacle density. *Discrete Comput. Geom.*, 20(4):561–587, 1998.

[183] S. Suri and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *Proc. ACM Symp. Comput. Geom.*, pp. 14–23, 1986.

[184] H. Sutanto and R. Sharma. Practical motion planning with visual constraints. In *IEEE Int. Symp. Ass. Task Plan.*, pp. 237–242, 1997.

[185] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker. A characterization of ten hidden-surface algorithms. *Comut. Surveys*, 6:1–25, 1974.

[186] S. J. Teller. Computing the antipenumbra of an area light source. *Comput. Graph.*, 26(2):139–148, 1992.

[187] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002.

[188] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Art. Intell.*, 128(1-2):99–141, 2000.

[189] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[190] J. Urrutia. Art gallery and illumination problems. Ch. 22 in *Handbook of Computational Geometry*, J.-R Sack and J. Urrutia (Eds.), Elsevier Science Publishers B.W., pp. 973–1026, 2000.

[191] US Geological Survey. `http://www.usgs.gov`

[192] J. Vleugels. *On Fatness and Fitness — Realistic Input Models for Geometric Algorithms*. Ph.D. Thesis, Universiteit Utrecht, 1997.

[193] C. Wang and B. Zhu. Three-dimensional weak visibility: Complexity and applications. *Theor. Comput. Sci.*, 234:219–232, 2000.

[194] R. Wein, J. P. van den Berg, and D. Halperin. The visibility-Voronoi complex and its applications. *Comput. Geom. Theory Appl.*, 36(1):66–87, 2007.

[195] E. Welzl. Constructing the visibility graph for n line segments in $O(n^2)$ time. *Inform. Proc. Lett.*, 20:167–171, 1985.

[196] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. *IEEE Comput. Graph. Appl.*, 10(6):13–32, 1990.

[197] T.-C. Yang and C.-S. Shin. Guard sufficiency set for polygons. *Journal of Korean Information Science and Technology*, 28(1–2):73–79, 2001.

[198] D. C. K. Yuen and B. A. MacDonald. Vision-based localization algorithm based on landmark matching, triangulation, reconstruction and comparison. *IEEE T. Robot.*, 21(2):217–226, 2005.

[199] Z. Zhang and O. Faugeras. A 3D world model builder with a mobile robot. *Int. J. Rob. Res.*, 11(4):269–285, 1992.

Berekenen en complexiteit van zichtbaarheid in geometrische omgevingen

Dit proefschrift beschrijft de oplossingen van een aantal zichtbaarheidsproblemen in geometrische omgevingen. Wat kun je zien vanuit een bepaald punt of gebied? Uit hoeveel stukken bestaat het zichtbare deel van de omgeving? En hoeveel tijd kost het om dit alles te berekenen?

Het onderzoek dat hier beschreven is, valt binnen de theoretische informatica. In het bijzonder valt het in het gebied van de computationele geometrie, waarin men algoritmen en data structuren ontwikkelt voor problemen die betrekking hebben op geometrische objecten zoals punten, lijnen, driehoeken, rechthoeken, kubussen en bollen. Theoretisch informaticaonderzoek is vaak fundamenteel en erg abstract. De problemen die worden bestudeerd zijn weliswaar afgeleid van concrete problemen uit andere vakgebieden of industriële toepassingen, maar door de sterke mate van theoretische abstractie is de achterliggende noodzaak van het vinden van een oplossing regelmatig onduidelijk. Daarom is het juist in dit vakgebied belangrijk om stil te staan bij het hoe en waarom van wat men onderzoekt, en dat doen we dan ook uitgebreid in het eerste hoofdstuk van dit proefschrift. We noemen hierin de drie belangrijkste toepassingsgebieden: de robotica, geografische informatiesystemen, en computer graphics. In de robotica is het vooral belangrijk om zo snel mogelijk te kunnen rekenen: robots moeten direct kunnen reageren op zichtbaarheidsinformatie die ze doorkrijgen van hun vaak simpele visuele sensors. In geografische informatiesystemen wordt ook veel gerekend aan zichtbaarheid, bijvoorbeeld met betrekking tot horizonsvervuiling bij het plannen van nieuwbouw. In computer graphics zijn schaduwberekeningen essentieel. Een schaduw die een lichtbron veroorzaakt is nauw verwant aan het niet-zichtbare gebied voor een waarnemer, en zichtbaarheid is dan ook nauw verwant aan (virtuele) verlichting.

In de computationele geometrie is er veel onderzoek gedaan naar zichtbaarheidsberekeningen. Het zogenaamde kunstgalerieprobleem (*Art Gallery problem* in het Engels) wordt algemeen beschouwd als het eerste zichtbaarheidsprobleem in de computationele geometrie. In dit probleem is de plattegrond van een museum gegeven, en wil men weten wat het minimum aantal camera's is dat er opgehangen moet worden

om alle schilderijen of andere kuntschatten te kunnen zien, en dus bewaken. In het tweede hoofdstuk doen we iets wat hier aan gerelateerd is. Voor een gegeven plattegrond proberen we een aantal ijkpunten te vinden die als het ware het museum discretiseren. Met behulp van deze ijkpunten kunnen we dan voor een gegeven verzameling camera's snel berekenen of deze camera's het hele museum zien. Hiertoe leiden we voorwaardes voor deze ijkpunten af, en ontwikkelen we een algoritme om te berekenen of een verzameling ijkpunten afdoende is. Helaas blijkt dit niet mogelijk voor alle mogelijke museumplattegronden: sommige musea blijken een oneindig aantal van deze ijkpunten nodig te hebben.

Hoofdstuk 2 is het enige hoofdstuk dat zich bezighoudt met zichtbaarheid in twee dimensies. Vanaf het derde hoofdstuk begeven we ons in de driedimensionale ruimte. Een belangrijk concept dat vaak terugkomt in dit proefschrift is een terrein: een verzameling aaneengesloten driehoeken die samen een (benadering van een) stuk van het aardoppervlak voorstellen. In het derde hoofdstuk berekenen we zichtbaarheid tussen twee gebieden van driehoeken op zo'n terrein. We doen dit voor drie varianten van zichtbaarheid: de twee gebieden moeten elkaar volledig zien, het ene moet minstens één punt bevatten van waaruit het andere gebied helemaal zichtbaar is en vice versa, of de twee gebieden zijn gedeeltelijk zichtbaar voor elkaar. We ontwikkelen efficiënte algoritmen voor al deze drie problemen.

De zichtbaarheidskaart (*visibility map* in het Engels) is de onderverdeling van de omgeving in zichtbare en onzichtbare delen. De complexiteit hiervan hangt af van zowel de complexiteit van de waarnemer als de complexiteit van de omgeving. In Hoofdstuk 4 leiden we boven- en ondergrenzen af voor de complexiteit van verschillende soorten zichtbaarheidskaarten. We bekijken twee soorten waarnemers — lijnsegmenten en driehoeken — evenals twee verschillende soorten omgevingen, namelijk terreinen en algemene verzamelingen driehoeken in 3D. De ondergrenzen die we afleiden stemmen voor deze laatste omgevingen vrijwel overeen met de bovengrenzen, terwijl er voor terreinen nog een niet te verwaarlozen kloof resteert. We presenteren ook efficiënte algoritmes om alle genoemde kaarten te berekenen.

Een van de kritiekpunten op ondergrensconstructies als die in Hoofdstuk 4 is dat ze nogal vergezocht zijn en in de praktijk niet snel voorkomen. Om dit kunstmatige verschil in complexiteit tussen theorie en praktijk te overbruggen, zijn realistische invoermodellen ontwikkeld. In zo'n model stel je een aantal aannames voor waaraan de invoer van je probleem moet voldoen, en met behulp van deze aannames kun je dan realistischere boven- en ondergrenzen bewijzen, of in het geval van algoritmes, lagere rekentijden. In Hoofdstuk 5 stellen we een realistisch invoermodel voor terreinen voor. We bewijzen dat de zichtbaarheidskaart minder complex is wanneer een terrein aan ons model voldoet. Zo'n lagere complexiteit bewijzen we ook voor twee datastructuren die met afstanden op terreinen te maken hebben: het kortstepad-netwerk en het Voronoi-diagram. Het invoermodel leidt in alle gevallen ook tot snellere algoritmes om de structuren te bouwen.

Als je een invoermodel realistisch noemt, moet je ook een onderbouwing geven waarom je van mening bent dat de invoer uit de praktijk voldoet aan de aannames in jouw model. In Hoofdstuk 6 voeren we een aantal experimenten uit, waarmee we

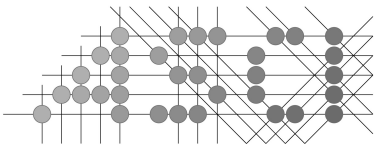
willen aantonen dat de aannames van het model uit Hoofdstuk 5 realistisch zijn. Met invoergegevens die we uit een geografisch informatiesysteem hebben verkregen, creëren we zestien terreinen, op verschillende niveaus van detail. We evalueren de waardes die de parameters van het invoermodel in deze terreinen hebben. In het bijzonder mogen deze waardes niet afhangen van het aantal driehoeken in het terrein, en dit lijkt ook niet het geval. Op deze manier verzamelen we bewijs voor de stelling dat het invoermodel inderdaad realistisch is. Tenslotte bekijken we de complexiteit van de zichtbaarheidskaart in deze zestien terreinen. Hierbij geven de resultaten aanleiding om te geloven dat de complexiteit inderdaad in de praktijk een stuk lager is dan in theorie mogelijk zou zijn.

We sluiten af met de opmerking dat het bij theoretisch onderzoek belangrijk is om de koppeling met de realiteit, oftewel de toepassingen, te behouden. In het ideale geval wordt er gestreefd naar er een continue wisselwerking tussen de ontwikkeling van theoretische beschrijvingen van de realiteit en experimentele verificatie van deze theorieën.

Samenvatting

Curriculum vitae

Esther Moet was born on February 28, 1980 in Dirksland, The Netherlands. In 2003, she received her Master's degree in computer science (cum laude) from the Rijksuniversiteit Groningen. A few months later, she started her Ph.D. studies at the Universiteit Utrecht in the group of Mark Overmars. In 2008, she completed this thesis there.



This work was carried out in the ASCI graduate school.
ASCI dissertation series number 162.