On the Minkowski sum of a terrain and a sphere

Javgal, Prithvi Subramanya

Award date:
2006

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

On the Minkowski Sum of a Terrain and a Sphere

by

Prithvi Subramanya Javgal

Supervisor:
prof. dr. Mark de Berg (TU/e)

Review Committee:
prof. dr. Mark de Berg
dr. Herman Haverkort
prof. dr. ir. Jack van Wijk

Eindhoven, November 2006

God made all the integers, the rest is man's doing.
Leopold Kronecker

Abstract

In this paper we study the Minkowski sum of a three-dimensional polyhedral terrain and a sphere in $\Re^3$. We first look into the two-dimensional equivalent of this problem. For this case we will show that the complexity of the Minkowski sum is linear and arrive at exact bounds for it. We present an algorithm that computes this Minkowski sum in linear time. In the three-dimensional case we show several results regarding the geometry of the Minkowski sum of a terrain and a sphere, and we present an algorithm which computes this in time $O(n^{2+\epsilon})$, for any $\epsilon > 0$. We also study the problem of computing the Minkowski sum of a piecewise-linear non self-intersecting finite arc in $\Re^2$ with a disk. An algorithm that computes this Minkowski sum in linear time is presented.

Acknowledgements

I am most grateful to prof. dr. Mark de Berg, my thesis advisor. Studying under his supervision has been a creative adventure with absolutely no dull moments. Without his excellent guidance and generous help this thesis would not have progressed far. The numerous valuable discussions that we had on the subject matter has formed a major part of this thesis. These discussions provided me with many insights into the skill of mathematical problem solving. Thanks to his several thorough reviews and reassessments, several mistakes and ambiguities from this document have been cleaned up which greatly improved the quality of this document. I am very grateful for the help and understanding that I received from him, given my situation of being a working part-time student. I feel very lucky to have studied under one of the pioneers in the field of Computational Geometry. The time that I spent on this thesis was academically most satisfying.

I would like to thank prof. dr. ir. Jan Friso Groote on behalf of all the teaching staff in the Computer Science Department and especially him for his help during the admissions to the degree program. In general all his staff in the Computer Science Department are very friendly and very helpful to all students.

I took up this course only after the encouragement of my parents. The support and encouragement of my parents, my brother and especially my wife Soumya, were the most invaluable ingredients that kept me going and enabled me to complete this course.

Prithvi S Javgal,
23rd October 2006,
Eindhoven, The Netherlands.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A three-dimensional terrain is a two-dimensional continuous surface in three-dimensional space with the special property that every vertical line intersects it at a single point if it intersects it at all. In a polyhedral terrain the two-dimensional surface is piecewise linear. More formally a three-dimensional *polyhedral* terrain can be defined as a mapping $f : S \subset \Re^2 \mapsto \Re^3$, with the special property that it is a piecewise linear and continuous function. From now on the set S is restricted to be a rectangle in $\Re^2$. For the sake of convenience and with a slight redefinition, we shall assume that the three-dimensional polyhedral terrain $\mathcal{T}$ consists not only of the two-dimensional surface (the graph of the function f) but also includes everything below that surface. It has the special property that any intersection of $\mathcal{T}$ with a vertical line is a single infinite line downwards.

Automated manufacturing using a robotic drill in CAD/CAM is a common method of producing well designed products. This involves creating the design using CAD software and using a so-called *milling machine* (also called CNC machine) to mill the actual prototype from styrofoam, plastic or other materials. The milling end of the robotic arm which, we call B, is called the *mill head*, and can be modeled as a ball of some fixed radius R, which is attached to the end of the robotic arm. Often the robotic arm itself can be moved only horizontally or vertically. This puts the restriction that only terrains can be milled by such a machine. From this perspective, once we have the construction of the terrain-like product finished using the designing software, the software controlling the robotic arm has to compute the *free configuration space* $\mathcal{C}_{\text{free}}(\mathcal{B}, \mathcal{T})$, which is the set of all points in $\Re^3$ where the robotic arm can move without damaging the construction. Computing this free configuration space when the designed product is a polyhedral terrain is an interesting problem and forms one of the motivations of this work.

Secondly the same problem also arises in geographical information systems, 3D game development, robot motion planning among others. In the robot motion planning problem, we have a spherical robot and the obstacle is a three-dimensional polyhedral terrain. Consider such a polyhedral terrain $\mathcal{T}$. The robot is represented by a spherical ball $\mathcal{B}$ of radius R. The *free configuration space* has to be found, which is a subset of $\Re^3$ in which any placement of the robot $\mathcal{B}$, does not intersect with the obstacle $\mathcal{T}$.

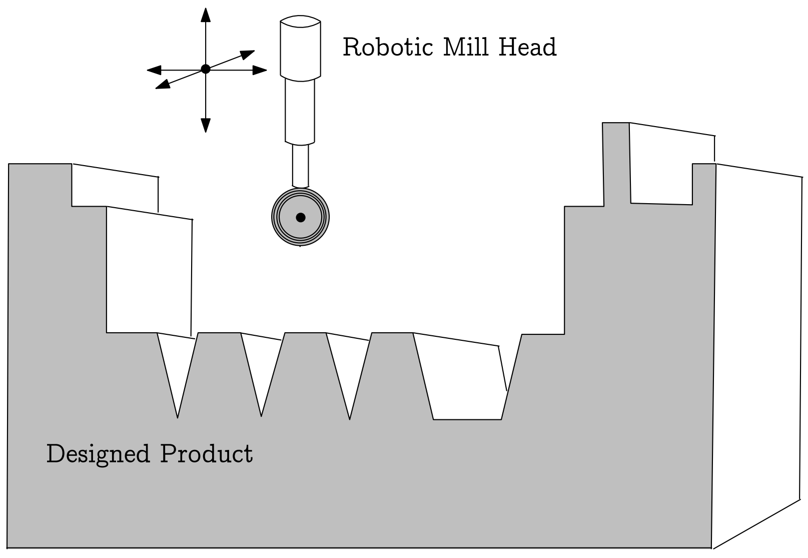

Figure 1.1: Producing 3D products with a milling machine.

Before tackling the three-dimensional problem we shall solve the two-dimensional equivalent in Section 2. The two-dimensional equivalent of the terrain is $\mathcal{T}$, which is a piecewise linear and continuous function in $\Re^2$, which satisfies the terrain property that any vertical line intersects it in at a single point if it intersects it at all. The circular robot shall be a disk $\mathcal{B}$ which is also of radius R.

In geographical information systems, it is interesting to know the *'buffer zone'* of a river or road network. This is the two strips of land on both sides of the river or road which are of a constant width and which follow the river or road during its course. The buffer-zone of a river is used in ecological research, agricultural research and other areas. The buffer zone of a road network is used for urban planning and analysis. Computing this buffer-zone for a given river or a stretch of road, and a specified width R of the buffer is similar to the problem of computing the free space in case of the two-dimensional terrain. The only difference here being that the river is not monotone with respect to any line unlike the terrain. Computing this buffer zone given the river modeled as a piecewise linear, non self-intersecting curve forms another goal of this work.

To illustrate the idea of free configuration space and its relation with Minkowski sums, consider the following example. Let B be a robot in $\Re^3$. The environment in which the robot B is allowed to move is called the *work space*. In the work space there is a set A of obstacles, which the robot has to avoid. A placement of the robot in the work space is also called a *configuration* of the robot. This placement is represented by a displacement vector from the origin in the work space. If the robot B is a translating rigid body, as we assume from now on, then the displacement vector is sufficient to specify a placement. Otherwise more parameters (eg. describing the orientation) are needed. More precisely if we fix a *reference point* inside the robot, then a displacement of the robot can be specified by giving the coordinates of the reference point. If the reference point is set at (x, y, z) then the robot is assumed to be placed such that, its reference point lies at (x, y, z). The *configuration space* is the set of all possible values of the reference point, which is the set of all possible displacements of the robot. Since in some placements the robot has a non-empty intersection with one or more of the obstacles, the set of all such placements is called the *forbidden configuration space*, denoted as $\mathcal{C}_{\text{forb}}(B, A)$. The rest of the configuration space, that is $\mathfrak{R}^3 \backslash \mathcal{C}_{\text{forb}}(B, A)$, is called the *free configuration space* $\mathcal{C}_{\text{free}}(B, A)$, wherein the robot does not intersect with any of the obstacles. The Minkowski sum of any two sets H_1 and H_2 in $\mathfrak{R}^3$ is defined as

$$H_1 \oplus H_2 = \{\vec{a} + \vec{b} \mid \vec{a} \in H_1 \text{ and } \vec{b} \in H_2\}.$$

It is an established fact [7], that $A \oplus -B$ is in-fact the forbidden space $\mathcal{C}_{\text{forb}}(B, A)$, that is, the set of placements of B which intersects with A. The complement of this is the free space - $\mathcal{C}_{\text{free}}(B, A)$.

In the problems that we study here, in the three-dimensional scenario the robot is a ball (a filled sphere) so $-B = B$, which is also true in the two-dimensional case of a disk (a filled circle). So in both cases $A \oplus B$ directly gives us the free configuration space. Consequently, in the sections that follow, we shall discuss the combinatorial complexities of $\mathcal{T} \oplus \mathcal{B}$ in both the two and three-dimensional cases, followed by algorithms for computing the Minkowski sums.

Previous Work

The Minkowski sums of several combinations of geometrical objects have already been studied in detail as shown in Table 1.1. We shall make use of some of the conclusions found in the listed works. In two-dimensions it is well known that the complexity of the union of m *pseudodiscs*[1] is $O(m)$ [15]. This property will be found to be useful in the two-dimensional case.

It has already been established in [3] that the combinatorial complexity of the Minkowski sum of a ball and a collection of polyhedral obstacles in $\mathfrak{R}^3$ with a total of n vertices is $O(n^{2+\epsilon})$ for any $\epsilon > 0$. Since $\mathcal{T}$ is a collection of n triangles, the same result on the upper-bound for the complexity holds for $\mathcal{T} \oplus \mathcal{B}$. In the same paper [3] the proof of the fact that the intersection of two *kreplach*[2] is a single continuous curve is also provided. This fact will be useful in the three-dimensional case.

An efficient algorithm for computing the Minkowski sum of a three-dimensional terrain and a closed convex polyhedron has been provided by Asano et al. in [5]. Once the polygonal faces of the terrain are triangulated, this work also describes a method defining an ordering of the triangular faces, which makes the design of the algorithm more convenient and more efficient. We use the same technique of ordering the triangular faces of the terrain in our algorithm.

[1]defined in Section 2 page 23

[2]explained in Section 4

Our Results

In Chapter 2 it is established that the combinatorial complexity for the two-dimensional problem of $\mathcal{T} \oplus \mathcal{B}$ is $O(n)$ where $\mathcal{T}$ is assumed to have n line segments. Although it is known that the complexity is linear for this problem, here we prove exact bounds for the complexity. Here the combinatorial complexity is defined as the number of half-lines, straight-line and circular-arcs that make up the Minkowski sum. Since this is always one more than the number of vertices in the Minkowski sum, the results will be stated as the number of vertices. An algorithm which constructs this Minkowski sum in $O(n)$ time is also presented.

In Chapter 3 it is established that the combinatorial complexity of the Minkowski sum of a disk with a finite, piecewise linear, non self-intersecting arc is linear. An algorithm that computes this Minkowski sum in linear time is presented.

In Chapter 4 for the three-dimensional version of the problem, we present many results on the geometry of $\mathcal{T} \oplus \mathcal{B}$. A conjecture regarding the combinatorial complexity of $\mathcal{T} \oplus \mathcal{B}$ based on its geometrical properties is given. The combinatorial complexity is taken to be the number of vertices in the Minkowski sum. An algorithm which constructs $\mathcal{T} \oplus \mathcal{B}$ in time $O(n^{2+\epsilon})$, for any $\epsilon > 0$ is also described.

Table 1.1: List of previous results for polygonal and polyhedral objects.

	B: Polygon/Polyhedron of complexity m **A**: Polygon/Polyhedron of complexity n				
2D	**B**	**A**	**Combinatorial Complexity of** $A \oplus B$	**Time Complexity of Algorithm**	**Ref.**
	Convex	Convex	$O(m+n)$	$O(m+n)$	[10]
	Convex	Monotone	$O(mn)$	$O(mn)$	[13]
	Convex	Simple	$O(mn)$	$O(mn\log(mn))$	[8]
	Disk	Simple	$O(n)$	$O(n)$	[15, 1]
	Star-shaped	Star-shaped	$O(mn(\min(m,n)))$	$O(mn\log(mn))$	[12, 16]
	Monotone	Simple	$O(mn^2)$	$O((mn+k)\log(mn))$, $k = O(mn^2)$	[13]
	Simple	Simple	$O(m^2n^2)$	$O(m\log(m) + n\log(n) + s + (s+k)\log(s) + k(m+n)\log(m+n))$, $k : O(m^2n^2)$, $s : O(mn)O(m^2n^2)$	[14]
3D	Convex	Convex	$O(mn)$	$O(mn)$	[11]
	Convex	k convex polyhedra	$O(nk\log(k))$ n: total complexity of the individual sums	randomized algorithm $O(nk\log(k)\log(n))$	[4]
	Ball	Set of n pairwise disjoint line segments in R^3	$O(n^{5/2+\epsilon})$ for any $\epsilon > 0$	randomized algorithm: $O(n^{5/2+\epsilon})$ for any $\epsilon > 0$	[2]
	Ball	Pairwise disjoint polyhedral obstacles in R^3 with a total of n vertices	$O(n^{2+\epsilon})$ for any $\epsilon > 0$	randomized algorithm: $O(n^{2+\epsilon})$ for any $\epsilon > 0$	[3]
	Convex	Polyhedral terrain with a convex projection	$O(n^2 m\log(n))$ n: number of faces of T m: number of faces of P	$O(nm+k+t)$ k: size of the output t: is at most $O(n^2 m\log(n))$	[5]

Chapter 2

The Two-Dimensional Case

2.1 Terminology and Notation

A two-dimensional polyhedral terrain is the graph of a piecewise linear continuous function $f : S \subset \Re \mapsto \Re$, where $S = [0, 1]$ is the domain of f. We shall assume that the two-dimensional polyhedral terrain $\mathcal{T}$ is composed of n line segments $s_1, s_2, \ldots, s_n$, ordered from left to right, excluding the two semi-lines at the boundaries. There are $n + 1$ vertices in the terrain, which we label as $v_0, v_1, \ldots, v_n$. The line segments make angles $\theta_1, \theta_2, \ldots, \theta_n$ respectively with the positive x-axis. The vertices of the segment s_i are v_{i-1} and v_i for $1 \leq i \leq n$. If we draw the two infinite vertical lines, $x = v_{i-1,x}$ and $x = v_{i,x}$, that pass through the end points of segment s_i as shown in Figure 2.2, then the area enclosed by these two vertical lines shall be denoted by $\Delta(s_i)$. In other words $\Delta(s_i) = [v_{i-1,x} : v_{i,x}] \cdot [-\infty : +\infty] = \{(x, y_1) | (x, y) \in s_i \text{ and } y_1 \in (-\infty, +\infty)\}$ where $v_{j,x}$ is the x-coordinate of v_j. The part of $\Delta(s_i)$ consisting of the line segment s_i and the area below it is a semi-infinite trapezoid which is denoted by $\bar{s_i}$. The two-dimensional polyhedral terrain $\mathcal{T}$ is the union of all the semi-infinite trapezoids $\bar{s_i}$, so $\mathcal{T} = \bigcup_i \bar{s_i}$. A part of the terrain will be denoted by $\mathcal{T}_{\leq a} = \bigcup_{1 \leq i \leq a} \bar{s_i}$. $\mathcal{T}_{>a}$ and $\mathcal{T}_{\geq a}$ are also used.

$\mathcal{B}$ is a disk of radius R centered at the origin. We use $\partial(o)$ to denote the boundary of the object o. $\partial_u(o)$ is the upper part of $\partial(o)$ of the convex object o. $\partial_d(o)$ is the lower part of $\partial(o)$. The Minkowski sum of the segment s_i with $\mathcal{B}$ is c_i, that is $c_i = s_i \oplus \mathcal{B}$. Similar to the definition of c_i, M_i is the Minkowski sum of $\bar{s_i}$ and $\mathcal{B}$, $M_i = \bar{s_i} \oplus \mathcal{B}$. We will use the notation of $M_{\leq a}$ to denote the union of a collection of individual M_i's. So $M_{\leq a} = \bigcup_{1 \leq i \leq a} M_i$. $M_{>a}$ and $M_{\geq a}$ are also similarly used.

2.2 Basic Properties

Consider the Minkowski sum of a single line segment s_i with a disk of radius R centered at the origin. This Minkowski sum is the union of a rectangle whose length is the length of s_i and whose height is $2R$ along with two disks of radius R centered at the end-points of s_i. Two sides of the rectangle are perpendicular to the line as shown in Figure 2.2.

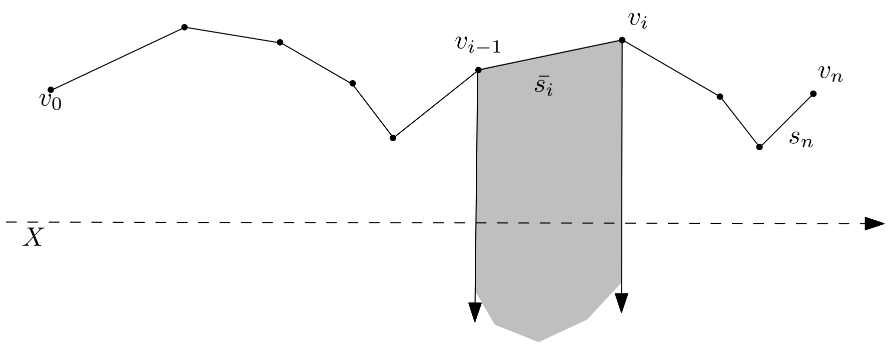

Figure 2.1: A two-dimensional terrain

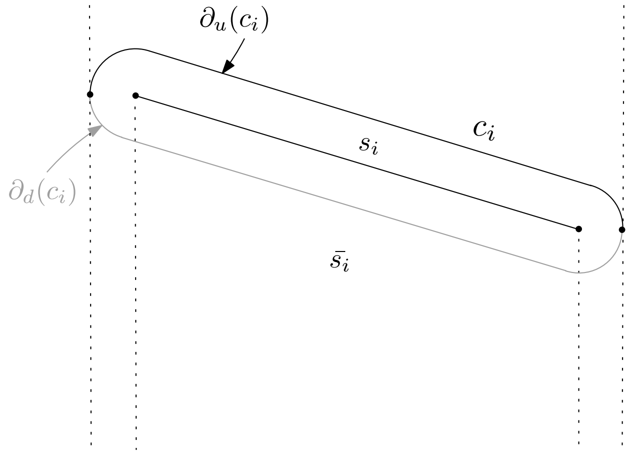

Figure 2.2: Cigar c_i with $\partial_u(c_i)$, $\partial_d(c_i)$ and $\bar{s_i}$

We shall call this sum a *cigar*. The cigar that is formed by the line segment s_i is c_i, so $c_i = s_i \oplus \mathcal{B}$. The boundary of c_i is denoted as $\partial(c_i)$. Any point on $\partial(c_i)$ would be the result of the sum of two vectors $\vec{a}$ and $\vec{b}$ where $\vec{a}$ is a point on the line segment s_i and $\vec{b} \in \partial(\mathcal{B})$.

Observation 2.2.1 *From every point of* s_i *the shortest distance to* $\partial(c_i)$ *is* R.

Given any c_i there are exactly two points which lie on its boundary $\partial(c_i)$, whose x-coordinates have the minimum and maximum values for all points in c_i. These two points partition the closed curve $\partial(c_i)$ into two parts. We shall call the upper part $\partial_u(c_i)$ and the lower part $\partial_d(c_i)$. Any vertical ray from $y = +\infty$ would intersect $\partial_u(c_i)$ before $\partial_d(c_i)$, if it intersects c_i at all.

From the definition of a polyhedral terrain the following observation can be concluded.

Observation 2.2.2 *The interior of the region* $\Delta(s_i)$ *does not contain any vertex* v_j *of* $\mathcal{T}$.

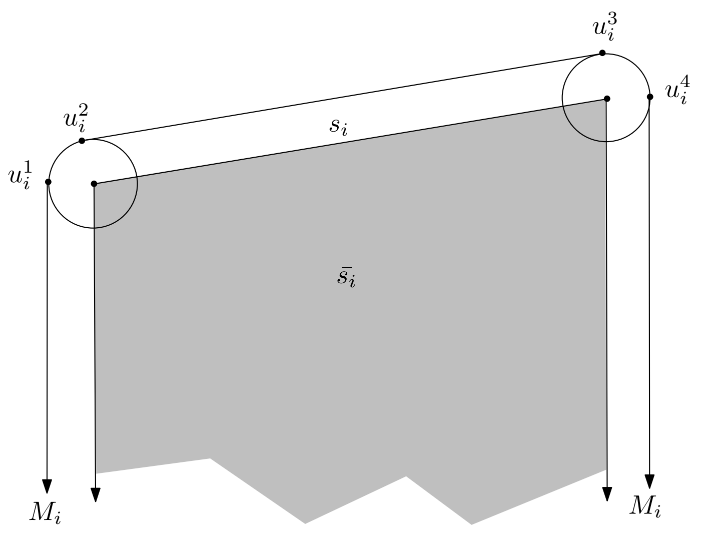

Figure 2.3: Definition of M_i

Recall that M_i is the Minkowski sum of $\bar{s_i}$ and $\mathcal{B}$, $M_i = \bar{s_i} \oplus \mathcal{B}$. Each M_i is a semi-infinite cigar that is obtained by Minkowski sum of $\bar{s_i}$ and $\mathcal{B}$, therefore we shall call each M_i a *semi-cigar* from now on. One such semi-cigar is shown in Figure 2.3.

Lemma 2.2.3 *$\mathcal{T} \oplus \mathcal{B}$ is a terrain and $\partial(\mathcal{T} \oplus \mathcal{B})$ consists of two half lines, and a number of line segments and circular arcs.*

Proof. We have to show that the intersection of any vertical line with $\mathcal{B} \oplus \mathcal{T}$ is a ray infinite downwards. Therefore it is sufficient to show that for some point $\vec{p} = (x, y) \in \mathcal{T} \oplus \mathcal{B}$ all other points $\vec{p_d} = (x, y_d)$ such that $y_d \leq y$ also belong to $\mathcal{T} \oplus \mathcal{B}$.

Assume that point $\vec{p} = (x, y) \in \mathcal{T} \oplus \mathcal{B}$. There must be points $\vec{a} = (x_a, y_a) \in \mathcal{T}$ and $\vec{b} = (x_b, y_b) \in \mathcal{B}$ such that $\vec{a} + \vec{b} = \vec{p}$. This implies $x_a + x_b = x$ and $y_a + y_b = y$. Now assume that there is an arbitrary point $\vec{p_d} = (x, y_d)$ directly below $\vec{p}$ such that $y_d \leq y$. Choose a point $\vec{a_d}$ that is directly below $\vec{a}$. Since all points which are directly below $\vec{a}$ also belong to $\mathcal{T}$, $\vec{a_d}$ must also belong to $\mathcal{T}$ because of the terrain property. Let $\vec{a_d} = (x_a, y_d - y_b)$. Since $y_d \leq y$ it must hold that $y_d - y_b \leq y - y_b$, we already know that $y - y_b = y_a$ so $y_d - y_b \leq y - y_b = y_a$. This means that the choice of the y-coordinate of $\vec{a_d}$ is right and it is indeed directly below $\vec{a}$. To continue observe that

$$\vec{a_d} \in \mathcal{T} \text{ and } \vec{b} \in \mathcal{B} \text{ resulting in } \vec{a_d} \oplus \vec{b} = (x_a + x_b, y_d) = \vec{p_d} \in \mathcal{T} \oplus \mathcal{B}.$$

So we have shown that any arbitrary point $\vec{p_d}$ below $\vec{p}$ also belongs to $\mathcal{T} \oplus \mathcal{B}$. Therefore it can be concluded that $\mathcal{T} \oplus \mathcal{B}$ is a terrain.

Each boundary of a semi-cigar is composed of two circular arcs, a line segment and two half lines. Therefore the boundary of the union of several semi-cigars, will be composed of parts of the circular arcs, line segments and half lines. □

The boundary of each M_i, denoted as $\partial(M_i)$, is also defined by four *implicit* vertices $I(M_i) = \{u_i^1, u_i^2, u_i^3, u_i^4\}$. These are the meeting points of the straight and circular segments of $\partial(M_i)$. Here $u_i^2 = \vec{v_{i-1}} + R\mathbf{n}$ and $u_i^3 = \vec{v_i} + R\mathbf{n}$, where $\mathbf{n}$ is a unit vector which has the direction of s_i after rotating it by $+90^o$. The points u_i^1 and u_i^4 are defined as $u_i^1 = \vec{v_{i-1}} - R\mathbf{i}$ and $u_i^4 = \vec{v_i} + R\mathbf{i}$. M_i is bounded by two semi-infinite vertical lines with one of their ends at u_i^1 and u_i^4 respectively. It also has two circular segments $\overline{u_i^1 u_i^2}$ and $\overline{u_i^3 u_i^4}$ which could possibly be empty in some cases. M_i always has a straight line edge $\overline{u_i^2 u_i^3}$ which is parallel to s_i and is of the same length. One such M_i is illustrated in Figure 2.3. The boundary of M_i, $\partial(M_i)$, is the union of the segments $\overline{u_i^1 u_i^2}$, $\overline{u_i^2 u_i^3}$, $\overline{u_i^3 u_i^4}$ and the two semi-infinite vertical lines at u_i^1 and u_i^4. The upper part of $\partial(M_i)$ is $\partial_u(M_i)$, which is the union of $\overline{u_i^1 u_i^2}$, $\overline{u_i^2 u_i^3}$ and $\overline{u_i^3 u_i^4}$. In fact $\partial_u(M_i) = \partial_u(c_i)$. The union of the n individual semi-cigars is

$$M_{\leq n} = \bigcup_{1 \leq i \leq n} M_i = \mathcal{T} \oplus \mathcal{B}.$$

The upper envelope of $\bigcup_{1 \leq i \leq n} \partial_u(M_i)$ is the same as the upper envelope of $\mathcal{T} \oplus \mathcal{B}$. Therefore:

$$\partial_u(\mathcal{T} \oplus \mathcal{B}) = \partial_u(M_{\leq n}).$$

From now on we will deal only with $\partial_u(M_{\leq n})$. We are interested in knowing the combinatorial complexity of $\partial_u(M_{\leq n})$ and an algorithm to compute it. The combinatorial complexity would be number of half-lines, straight-line segments and circular-arcs that make up $\partial_u(M_{\leq n})$. The algorithm is discussed in Section 2.4. To find the combinatorial complexity, the approach will be to count the number of vertices in $\partial_u(M_{\leq n})$, by knowing this it is easy to count the number of segments since each vertex is adjacent to exactly two segments of $\partial_u(M_{\leq n})$. The idea of the implicit vertex has already been introduced. We define *explicit* vertices of $\partial_u(M_{\leq n})$ as those vertices in $\partial_u(M_{\leq n})$ that are not implicit vertices of some cigar. Clearly the total number of vertices would then be sum of the number of implicit and explicit vertices. An explicit vertex is formed when two non-adjacent semi-cigars intersect. They are also formed between adjacent semi-cigars, M_i and M_{i+1} if $\theta_i - \theta_{i+1} < \pi$.

The implicit vertex u_i^3 will be present in $\partial_u(M_{\leq n})$ if $\theta_i - \theta_{i+1} \geq \pi$. Similarly if $\theta_{i-1} - \theta_i \geq \pi$ then u_i^2 will be present in $\partial_u(M_{\leq n})$. Since both these conditions can be true for a particular M_i, it can be concluded that each M_i contributes at most 2 implicit vertices to $\partial_u(M_{\leq n})$. The two implicit vertices of a semi-cigar mentioned here, is not shared with adjacent semi-cigars. Similarly if $\theta_i - \theta_{i+1} < \pi$ then the implicit vertex u_i^3 will not be part of $\partial_u(M_{\leq n})$ and if $\theta_{i-1} - \theta_i < \pi$ then the implicit vertex u_i^2 will not be part of $\partial_u(M_{\leq n})$. To summarize we have the following observation.

Observation 2.2.4 *Each semi-cigar M_i, contributes*

- *A maximum of 2 implicit vertices to $\partial_u(M_{\leq n})$.*

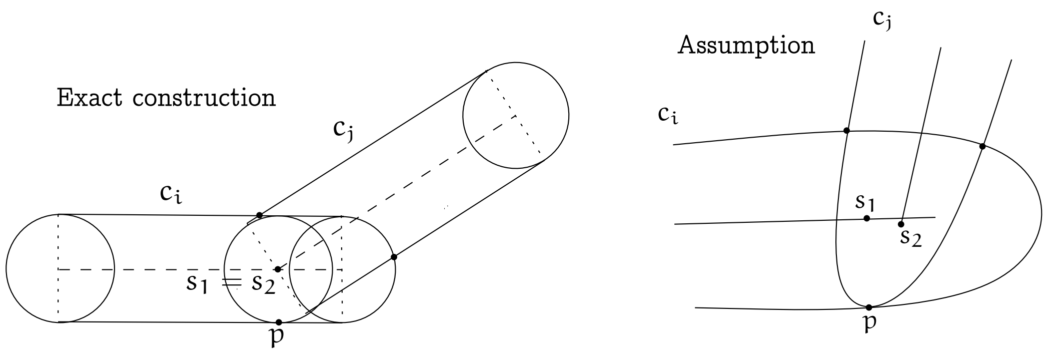

Figure 2.4: Intersection of c_i and c_j such that they violate the pseudodisc property

- *A minimum of 0 implicit vertices to* $\partial_u(M_{\leq n})$.

Lemma 2.2.5 *For* $i \neq j$, *for non-adjacent cigars* c_i *and* c_j, $\partial(c_i) \cap \partial(c_j)$ *has at most two intersection points.*

Proof. Let c_i and c_j be two non-adjacent cigars. Notice that any cigar c_i is convex for all i. Assume that $\partial(c_j)$ intersects $\partial_u(c_i)$ at two points and touches $\partial_d(c_i)$ at a single point. Let the point of contact of $\partial(c_j)$ and $\partial_d(c_i)$ be point $\vec{p}$, as shown in Figure 2.4. Rotate the axes such that $\vec{p}$ has the lowest y-coordinate value of all points in c_i and c_j. In this position we can express $\vec{p}$ in two ways as the Minkowski sum of a point $\vec{s_1}$ in c_i or as the Minkowski sum of a point $\vec{s_2}$ in c_j, so

$$\vec{p} = \vec{s_1} + \vec{r_1} = \vec{s_2} + \vec{r_2}.$$

Since $\vec{p}$ is a point with the lowest y-coordinate, $\vec{r_1}$ and $\vec{r_2}$ must be uni-directional, pointing vertically down, thus $\vec{r_1} = \vec{r_2}$. This leads us to the conclusion that $\vec{s_1} = \vec{s_2}$ from the previous equation. This cannot be possible since we started with two non-adjacent cigars, so they cannot have common points. As a consequence of this contradiction, it can be concluded that the third intersection point $\vec{p}$ cannot exist. On the same lines the case of $\partial(c_j)$ intersecting $\partial_d(c_i)$ at two points and touching $\partial_u(c_i)$ at one point can be similarly proved to be impossible. From this the lemma follows. □

With the previous lemma it is already possible to make conclusions about the number of vertices in $\partial_u(M_{\leq n})$. The vertices in the output are either implicit or explicit. The maximum number of implicit vertices that can show up in the output is $2n + 2$. The minimum number of implicit vertices that can show up in the output is 2. These are illustrated in the constructions described in Figure 2.7 and Figure 2.8. For each cigar we have $n - 2$ non-adjacent cigars in the terrain that can possibly intersect with it to produce an explicit vertex.

A pair of planar objects a_1 and a_2 are called pseudodiscs if they satisfy the pseudodisc property that the sets $a_1 \backslash a_2$ and the set $a_2 \backslash a_1$ are connected . For m pseudodiscs in the

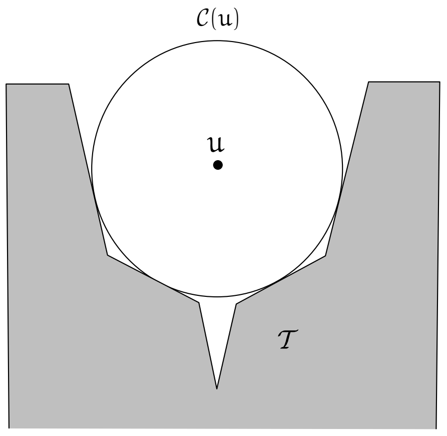

Figure 2.5: Unique circle of 2 or more segments

plane, it is known that the complexity of their union is $O(m)$. In our case it is equivalent to having $n-2$ pseudodiscs since for any cigar there are at most $n-2$ non-adjacent cigars. So we can conclude that the number of explicit vertices in the Minkowski sum is bounded by $O(n)$. As a consequence the total number of vertices in the Minkowski sum and hence the combinatorial complexity is bounded by $O(n)$.

Observation 2.2.6 *For each vertex u of $\partial_u(M_{\leq n})$, there is unique circle of radius R, denoted $\mathcal{C}(u)$, centered at u, which lies completely above $\mathcal{T}$ and touches at least 2 segments of $\mathcal{T}$.*

This is direct consequence of the definition of the Minkowski sum. Notice that the circle can touch $\mathcal{T}$ in more than two points, two being the minimum number of segments. This is shown in Figure 2.5.

2.3 Complexity of the Minkowski Sum

Lemma 2.3.1 *$\partial_u(M_{<i})$ and $\partial_u(M_i)$ intersect in at most one explicit intersection point.*

Proof. Let ν_{i1} be the left-most explicit vertex in the intersection of $\partial_u(M_{<i})$ and $\partial_u(M_i)$, illustrated in Figure 2.6. We will show that there cannot be any other explicit intersection point ν_{i2} between $\partial_u(M_{<i})$ and $\partial_u(M_i)$. Before continuing with the proof we need a few definitions. The unique circle corresponding to the vertex ν_{i1} is $\mathcal{C}(\nu_{i1})$, this circle touches the line segment s_i at the point p. The line l_2 is the vertical line through p and the line l_1 is the vertical line that touches the left-most point of $\mathcal{C}(\nu_{i1})$. The region to the left of l_1 and l_2 and outside $\mathcal{C}(\nu_{i1})$ is denoted by $\mathcal{S}$. This is indicated by the shaded region in the diagram.

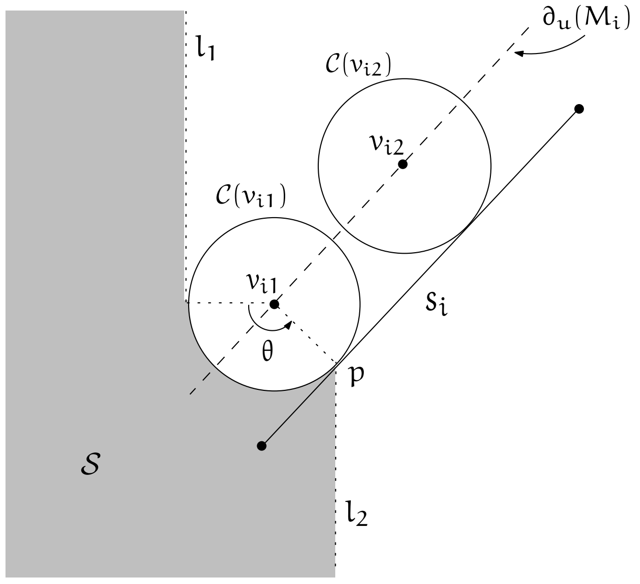

Figure 2.6: Illustration of proof of Lemma 2.3.1

Notice that $\mathcal{T}_{<i} = \bigcup_{1 \le j < i} s_j$ lies completely to the left of p. In-fact it lies completely to the left of $x(\nu_{i-1})$, because of the terrain property. Also none of the edges of $\mathcal{T}_{<i}$ can intersect the interior of $\mathcal{C}(\nu_{i1})$. Therefore $\mathcal{T}_{<i}$ lies in the shaded region $\mathcal{S}$. If ν_{i2} is a point to the right of ν_{i1}, and if $\partial_d(\mathcal{C}(\nu_{i2}))$ touches s_i, then $\mathcal{T}_{<i}$ cannot touch $\mathcal{C}(\nu_{i2})$. This implies that there cannot be an explicit intersection point ν_{i2} between $\partial_u(M_{<i})$ and $\partial_u(M_i)$, to the right of ν_{i1}. Therefore we conclude that $\partial_u(M_{<i})$ and $\partial_u(M_i)$ intersect in at most one explicit intersection point. □

Corollary 2.3.2 *$\partial_u(M_{>i})$ and $\partial_u(M_i)$ intersect in at most one explicit intersection point. Consequently each $\partial_u(M_i)$ contributes at most 2 explicit intersection vertices to $\partial_u(M_{\le n})$.*

From Corollary 2.3.2 we know that each c_i contributes at most two explicit vertices to $\partial_u(M_{\le n})$. Each explicit vertex is shared among the two cigars. So the maximum number of explicit vertices is $n-1$ for the n line segments. A situation that leads to the maximum number of explicit vertices is shown in Figure 2.7. In this construction, additionally the first and the last segments will add at least 2 implicit vertices each. So the total comes to $n+3$ vertices. However this is *not* the lower-bound on the combinatorial complexity of $\partial_u(M_{\le n})$. An example having only 3 vertices in $\partial_u(M_{\le n})$ is shown in Figure 2.7 to illustrate this.

Observation 2.3.3 *The maximum number of explicit vertices in $\partial_u(M_{\le n})$ is $n-1$.*

Theorem 2.3.4 *The combinatorial complexity of $\partial_u(M_{\le n})$ is at most $2n+2$ and this bound is tight in the worst case.*

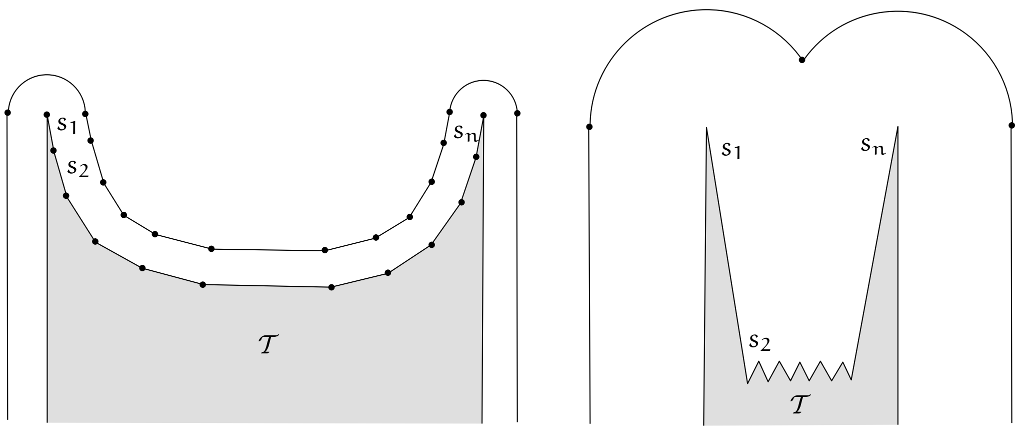

Figure 2.7: Constructions that give maximum number of explicit vertices and a lower-bound counter example.

Proof. Notice that each semi-cigar can contribute *individually* 2 implicit vertices to $\partial_u(M_{\leq n})$ giving a total of $2n$. The two cigars corresponding to the two half-lines at the two ends of the terrain can contribute one additional vertex each, corresponding to $v_0 - R\mathbf{i}$ and $v_n + R\mathbf{i}$. This gives a maximum total of $2n+2$. This gives an upper-bound on the combinatorial complexity. A situation that leads to the upper-bound on the combinatorial complexity is shown in Figure 2.8. In this construction notice that are no explicit vertices. Even if a single explicit vertex were to be present, then the terrain can be suitably deformed to produce two implicit vertices instead of one explicit vertex. Therefore to arrive at an upper-bound our construction should not have any explicit vertices. Therefore the bound is tight. Consequently total number of segments(straight, circular, semi-infinite) would be $2n+3$ for the upper-bound. □

2.4 Algorithm

The curve $\partial_u(M_i)$ is bounded by the edges between vertices u_i^1 and u_i^4, which are $\overline{u_i^1u_i^2}$, $\overline{u_i^2u_i^3}$ and $\overline{u_i^3u_i^4}$. Each M_i also has two semi-infinite vertical line segments starting at u_i^1 and u_i^4, which we shall denote as $\overline{u_i^1}$ and $\overline{u_i^4}$ respectively. So the edges of M_i correspond to the set $\{\overline{u_i^1}, \overline{u_i^1u_i^2}, \overline{u_i^2u_i^3}, \overline{u_i^3u_i^4}, \overline{u_i^4}\}$. Using this frame-work we shall look into Algorithm *Mink2D*, which finds the edges in $\partial(M_{\leq n})$. Algorithm *Mink2D* uses the incremental construction approach. Clearly,

$$\mathcal{B} \oplus \mathcal{T} = \bigcup_{1\leq i\leq n} M_i = (((M_1 \cup M_2) \cup M_3) \ldots) \cup M_n.$$

This suggests computing M_1 first and incrementally adding $M_2, M_3, \ldots, M_n$ to the union. At each iteration of the algorithm we will have a valid terrain. The edges are

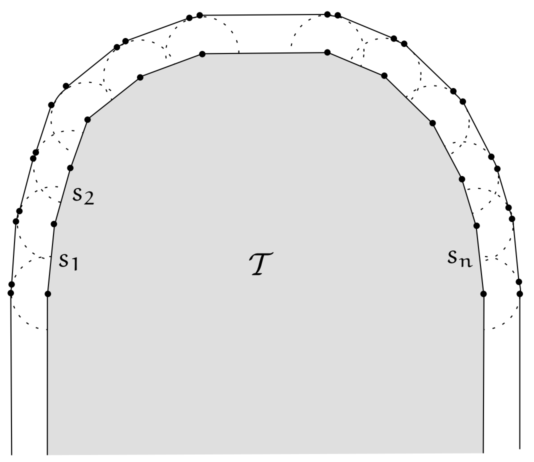

Figure 2.8: Construction that gives that shows the tight upper-bound.

stored in a list $\mathcal{M}$ in the correct order from left to right. In the first step of the algorithm, the edges of M_1 are added to $\mathcal{M}$. One by one, each M_i for $2 \le i \le n$ is computed, the intersection point of $\partial(M_i)$ with $\partial(M_{<i})$ is found, followed by construction of $\partial(M_{\le i})$. New edges of $\partial(M_i)$ are added to $\mathcal{M}$ and existing edges are modified to compute the union. This procedure is illustrated in Figure 2.9.

```
Algorithm Mink2D($\mathcal{T}$, R)
Output: A list $\mathcal{M}$ of edges of $\partial_u(M_{\le n})$, ordered from left to right.
1.  $\mathcal{M} \leftarrow M_1$
2.  for $i \leftarrow 2$ to $n$
3.       do (* Update $\mathcal{M}$ by computing $M_{<i} \cup M_i$ *)
4.          done $\leftarrow$ false
5.          while done is false
6.             do Remove the last curve $\gamma$ from $\mathcal{M}$
7.                if $\gamma$ is completely below $\partial(M_i)$
8.                   then skip
9.                   else Compute $\gamma' \leftarrow \gamma \backslash M_i$,
10.                       (* $\gamma'$ is the portion of $\gamma$ that lies outside $M_i$ *)
11.                       Append $\gamma'$ to $\mathcal{M}$
12.                       Append the portion of $\partial(M_i)$ that is to the right of $\gamma'$ to $\mathcal{M}$
13.                       done $\leftarrow$ true
14. return $\mathcal{M}$
```

From Lemma 2.3.1 we know that there is at most one explicit vertex at the intersection of $\partial(M_{<i})$ and $\partial(M_i)$, or alternatively if there is an overlap then we will have a single overlap between them. As a result Algorithm *Mink2D* makes the correct assumption of deleting all the edges of $\partial(M_i)$ that are to the left of the intersection-point or overlap

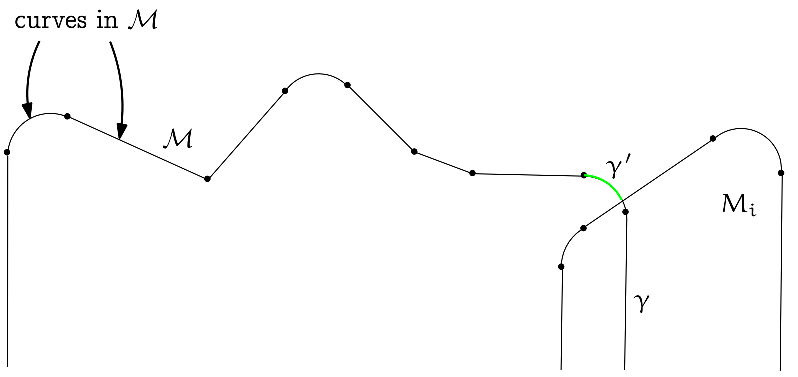

Figure 2.9: Illustration of Algorithm *Mink2D*

and all the edges that are to the right of the intersection-point or overlap from L. Hence the algorithm is correct.

Theorem 2.4.1 *Algorithm Mink2D computes* $\partial_u(M_{\leq n})$ *in* $O(n)$ *time.*

Proof. The outer for-loop is executed $n-1$ times for each of $M_2, M_3, \ldots, M_n$. However the inner while-loop may execute an arbitrary number of times for each M_i. The total time needed for the inner while loop is linear in the number of edges that are being discarded. Since in total at most 5 edges are added to $\mathcal{M}$ for each semi-cigar and each edge is discarded at most once, this means that the overall running time is still $O(n)$. □

Chapter 3

The Buffer-Zone Problem

3.1 Terminology and Notation

In this section we discuss the buffer-zone problem. A river or a tributary can usually be modeled as single curve from its source till the point where it meets a bigger river or the ocean. This curve is very rarely self-intersecting, in which case the point where the self-intersection takes place can be used to separate the river into three individual sub-rivers which are non self-intersecting. As a consequence of this we shall assume that the river is composed of the union of several sub-rivers which is a piecewise-linear, non self-intersecting continuous curve in $\Re^2$.

Figure 3.2, shows a river and its tributaries. The sequence of line segments joining the vertices $(v_3, v_8, v_9, v_{10}, v_{12}, v_7, v_{10}, v_{11})$ is the main river. The tributaries join this river at the specified vertices. The main river is self intersecting at the vertex v_{10}. So the main river is partitioned into two non self-intersecting sub-rivers $\mathcal{R}_1 = (v_3, v_8, v_9, v_{10}, v_{12})$ and $\mathcal{R}_2 = (v_{12}, v_7, v_{10}, v_{11})$. In this case v_{12} is an arbitrary point that was suitably chosen.The tributaries are also similarly divided into sub-rivers. The other sub-rivers in this case are $\mathcal{R}_3 = (v_2, v_6, v_7)$, $\mathcal{R}_4 = (v_1, v_6)$, $\mathcal{R}_5 = (v_4, v_8)$ and $\mathcal{R}_6 = (v_5, v_9)$. The entire river $\mathcal{R}$ is the union of these sub-rivers, so $\mathcal{R} = \bigcup_{1 \leq i \leq 6} \mathcal{R}_i$. Also notice that when this partitioning is done we can know the set of vertices at which the sub-rivers 'break up'. This set of vertices called the fusion vertices $F(\mathcal{R}) = \{v_6, v_7, v_{12}, v_8, v_9\}$. These will be useful at a later stage when the sub-rivers have to be fused back, to get the complete river.

Using this approach we shall assume that the entire river is represented by $\mathcal{R}$, this is individually composed of the sub-rivers so $\mathcal{R} = \bigcup_i \mathcal{R}_i$. From this point on the discussion will be mainly about one of these sub-rivers $\mathcal{R}_k$, to make things more convenient we shall drop the sub-script k and just use $\mathcal{R}$, since the discussion is valid for any sub-river. From the context it will be clear whether we are talking about a sub-river or the complete river. Without loss of generality let one of the sub-rivers be $\mathcal{R}$ and let it be composed of n line segments $s_1, s_2, \ldots, s_n$, ordered from source to drain. There are $n + 1$ vertices in this sub-river, which we label as $v_0, v_1, \ldots, v_n$. The line segments make angles $\theta_1, \theta_2, \ldots, \theta_n$ respectively with the positive x-axis. The vertices of the segment s_i are v_{i-1} and v_i for

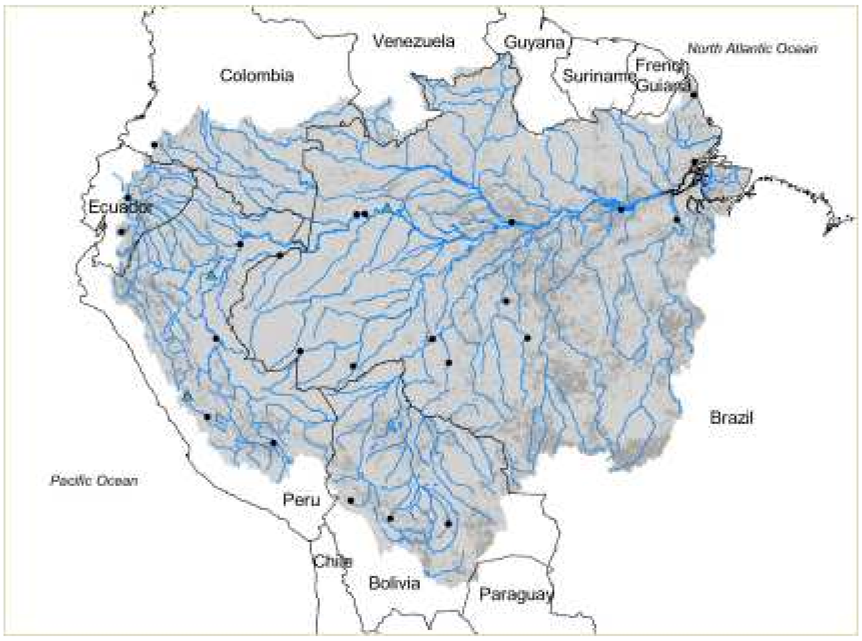

Figure 3.1: The Amazon and its tributaries [9].

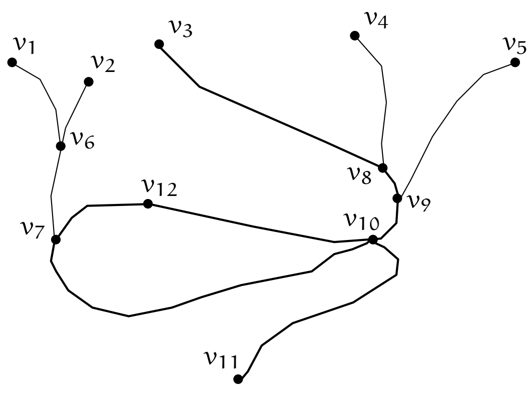

Figure 3.2: The definition of a sub-river.

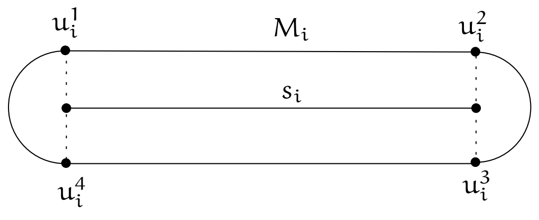

Figure 3.3: A cigar of the buffer-zone problem.

$1 \leq i \leq n$. The sub-river $\mathcal{R}$ is the union of all the line segments s_i, so $\mathcal{R} = \bigcup_i s_i$.

Like in the previous scenario of a two-dimensional terrain, the buffer-zone of a single segment s_i is the Minkowski sum of s_i with a ball $\mathcal{B}$ of radius R. Like before we shall call this sum a *cigar*. The cigar that is formed by the line segment s_i is M_i, that is $M_i = s_i \oplus \mathcal{B}$. The boundary of M_i is denoted as $\partial(M_i)$. Each cigar M_i has four implicit vertices, $I(M_i) = \{u_i^1, u_i^2, u_i^3, u_i^4\}$ and four edges joining these vertices as shown in Figure 3.3. Additionally we shall use the now familiar notation of $M_{\leq j} = \bigcup_{1 \leq i \leq j} M_i$, so that $M_{\leq n}$ is the buffer-zone of the entire sub-river $\mathcal{R}$.

3.2 Bounding the Complexity

Given a single sub-river $\mathcal{R}$ with n line segments and a disc of radius R, we are interested in knowing the complexity of the boundary of the buffer-zone $\partial(M_{\leq n})$. We already know that the complexity of the union of n pseudodiscs in $\mathfrak{R}^2$ is $O(n)$. There are n individual cigars that are formed by the Minkowski sum of the n segments and the disc of radius R. These n cigars pairwise satisfy the pseudodisc property because Lemma 2.2.5 still holds true for this case. This is due to the fact that the single sub-river $\mathcal{R}$ is non self-intersecting. Using the pseudo-disc property it follows that the bound on the complexity of the boundary of the union of the cigars is $O(n)$.

3.3 A Line Sweep Algorithm

In order to compute the buffer-zone of the complete river the buffer-zones of the sub-rivers are individually computed first. The union of these Minkowski sums are then computed to get the buffer-zone of the entire river. So the most important problem is to compute the buffer-zone of a single sub-river $\mathcal{R}$. The curve $\overline{\partial(M_i)}$ is bounded by the edges between vertices u_i^1, u_i^2, u_i^3 and u_i^4, which are $\overline{u_i^1 u_i^2}$, $\overline{u_i^2 u_i^3}$, $\overline{u_i^3 u_i^4}$ and $\overline{u_i^4 u_i^1}$. Using this frame-work we shall look into Algorithm *SubRiverBufferZone*, which finds the edges in $\partial(M_{\leq n})$. Clearly,

$$\mathcal{B} \oplus \mathcal{R} = \bigcup_{1 \leq i \leq n} M_i = (M_1 \cup M_2) \cup (M_2 \cup M_3) \cup \ldots \cup (M_{n-1} \cup M_n).$$

The first major step Algorithm *SubRiverBufferZone* is to construct a partial union of the n cigars. That is the union of each cigar M_i with *only* its adjacent neighbors. To illustrate, M_1 and M_2 are first computed followed by the union of M_1 and M_2. Next M_3 is computed followed by the union of M_2 and M_3. This is continued till all the cigars are linked with their adjacent neighbors. The only problem that may arise is that the current cigar M_i that is being added may intersect with a non-adjacent cigar M_j where $j < i$. This would mean that the overlapping regions of M_j and M_i would have to be removed. To achieve this efficiently, in the second major step of the algorithm, the line segment intersection algorithm [7] is used. The line segment intersection algorithm reports the intersection points of n line segments in $\Re^2$ in general position in $O(n \log n + k \log n)$ time, where k is the output size - the actual number of intersection points of the segments. Here we adapt this algorithm to compute the intersection points of $O(n)$ arcs which are either straight or semi-circular. Once an intersection point is known the current edges of the cigars which are involved in the intersection are modified to compute the union. It is possible to use the line segment intersection algorithm with a few modifications in our case since there are only 6 types of intersections possible between the straight and circular arcs of $M_{\leq n}$. These 6 types of intersections are shown in Figure 3.4.

The only disadvantage of this approach is that the line segment intersection algorithm computes all the intersection points in $M_{\leq n}$ which are $O(n^2)$ in number. This makes the algorithm highly inefficient. However notice that we are not interested in computing the vertices in the interior of $\partial(M_{\leq n})$ but only interested in the arcs and vertices on it. Therefore an additional step in the algorithm is introduced which removes some of the arcs and vertices of M_i and M_{i+1} which lie in the interior of $M_i \cup M_{i+1}$, which is the union of adjacent cigars M_i and M_{i+1}. This step of removing the unwanted arcs and vertices is done after the partial union of the adjacent cigars is computed and before using the line segment intersection algorithm. This improves the running time of the algorithm considerably. An example of the removal of redundant edges is shown Figure 3.5. Algorithm *SubRiverBufferZone* illustrates the algorithm discussed till now.

```
Algorithm SubRiverBufferZone($\mathcal{R}$, R)
Output: $M$ - the DCEL containing the edges and vertices of $\partial(M_{\leq n})$
1.  $M \leftarrow M_1$
2.  for $i \leftarrow 2$ to $n$
3.      do Compute $M_i$.
4.          Update $M$ with $M_{i-1} \cup M_i$.
5.          Remove redundant edges in the union $M_{i-1} \cup M_i$.
6.  Use the arc intersection algorithm to compute the intersection points of the arcs in
    $M$. Store all the intersection points in a list $L$.
7.  At each intersection point in $L$, compute the union of the cigars which are involved
    and update $M$.
8.  return $M$
```

Next we shall analyze the time complexity of Algorithm *SubRiverBufferZone*. The computation of a single cigar from its line segment takes $O(1)$ time. To compute the n

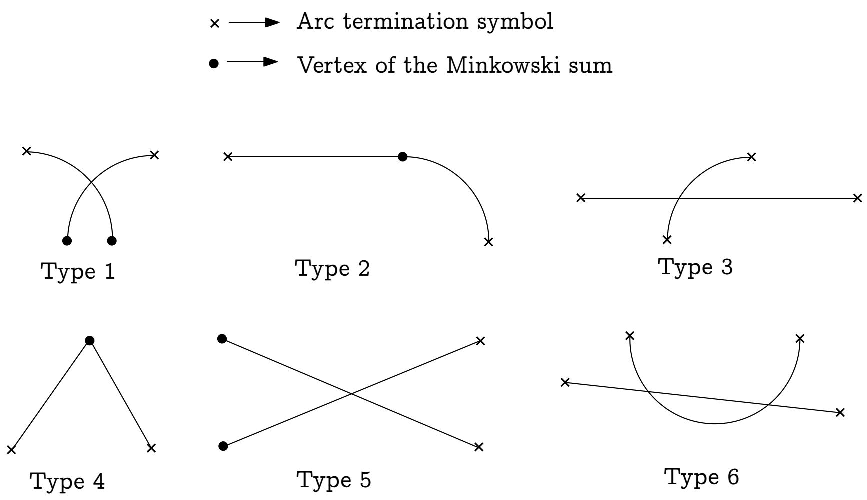

Figure 3.4: Types of arc intersections in $M_{\leq n}$.

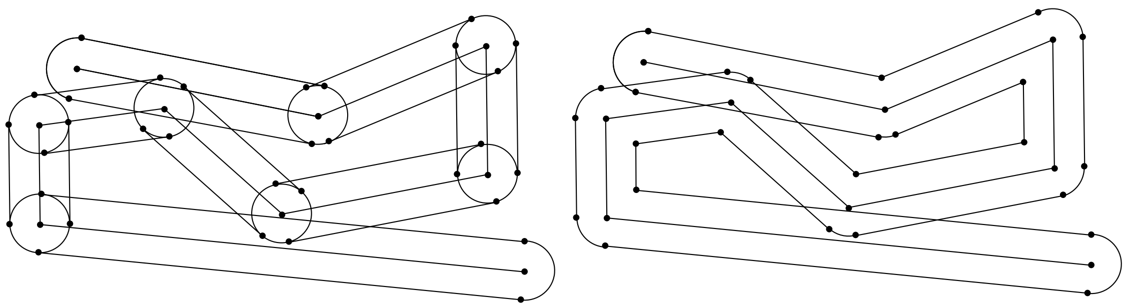

Figure 3.5: Removing redundant edges and vertices of adjacent cigars.

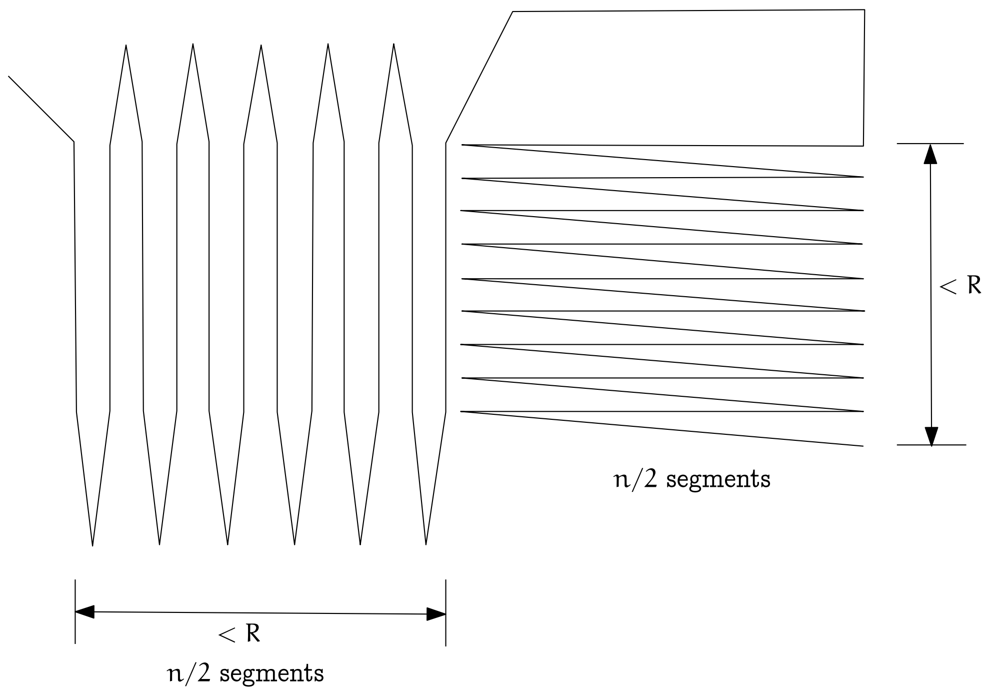

Figure 3.6: Sub-river showing a potential $O(n^2)$ intersections in $M_{\leq n}$.

cigars it takes $O(n)$ time. Computing the union of two adjacent cigars and removing the redundant vertices and arcs takes time proportional to the total number of vertices and arcs that are handled. Since the maximum number of vertices and arcs are fixed for any two cigars, this operation is also $O(1)$. This operation is done $n - 1$ times, resulting in $O(n)$ time complexity. The computation of the arc intersection using the line sweep algorithm from [7], takes $O(n \log n + k \log n)$ time where k is the number of intersections that are found. If we can estimate the value of k then the time complexity of the algorithm will follow. We know that the complexity of $\partial(M_{\leq n})$ is $O(n)$, so k must be at least $O(n)$. The number of intersections in the interior of $\partial(M_{\leq n})$ also contribute to k. Next we shall try to estimate the total number of such intersections.

Algorithm *SubRiverBufferZone* only removes the unwanted edges and vertices between adjacent cigars. It is possible that a cigar intersects a non-adjacent cigar in the sub-river. The algorithm also finds all the intersection-vertices of non-adjacent cigars that lie in the interior of $\partial(M_{\leq n})$. These vertices are not of interest to us since they lie in the interior of the buffer-zone. It can also be the case that the number of such intersection-vertices of non-adjacent cigars is quite large. Consider the sub-river shown in Figure 3.6. In this sub-river there are roughly $n/2$ segments forming a vertical pattern and another $n/2$ segments forming a horizontal pattern as shown in the figure. The width of the vertical pattern and the height of the horizontal patterns are both less that R. Notice that this sub-river has roughly $n/6$ non-adjacent vertical segments and roughly $n/4$ non-adjacent horizontal segments, let us label these segments as be-

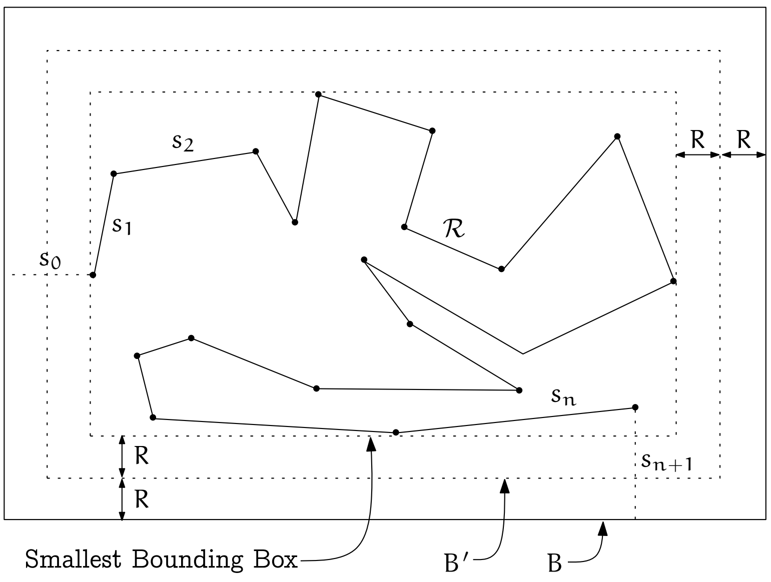

Figure 3.7: Defining two simple polygons from $\mathcal{R}$ using a bounding box.

longing to the sets V and H respectively. Notice that for $v \in V$ and for $h \in H$, the intersection of the cigars of segments v and h results in intersection-vertices in $M_{\leq n}$. There are $O(n^2)$ such intersections between cigars of V and H. All these vertices are computed by Algorithm *SubRiverBufferZone*. Although this a very rare scenario, theoretically it is possible and therefore it must be considered while estimating the worst case running time. Therefore k is $O(n^2)$, which means that the time complexity of Algorithm *SubRiverBufferZone* is $O(n^2 \log n)$. This can be considerably improved and a better algorithm can be designed as we shall see in the next section.

3.4 A Medial-Axis Based Algorithm

For a given sub-river $\mathcal{R} = \bigcup_{1\leq i\leq n} s_i$, clearly there is a unique *smallest* bounding box. We shall define another bounding box B of $\mathcal{R}$, whose center is at the same point as that of the smallest bounding box, and whose width and height are increased by 4R compared to the smallest bounding box, as shown in Figure 3.7. Using this bounding box and $\mathcal{R}$ it is possible to define two simple polygons. From the first and last vertices of $\mathcal{R}$, namely v_0 and v_n, two new segments s_0 and s_{n+1} are added to $\mathcal{R}$ such that $\bigcup_{0\leq i\leq n+1} s_i$ partitions B into two sets as shown in Figure 3.7. Clearly these two sets define a simple polygon each. In the example shown it is possible to just add two segments s_0 and s_{n+1} and partition the bounding box B into two simple polygons, it may however not be this simple for all sub-rivers. Special methods will have to be used for partitioning B into simple polygons in such cases, these will be discussed later in section 3.4.1.

The Voronoi diagram of a set of sites is a partition of the plane into connected faces,

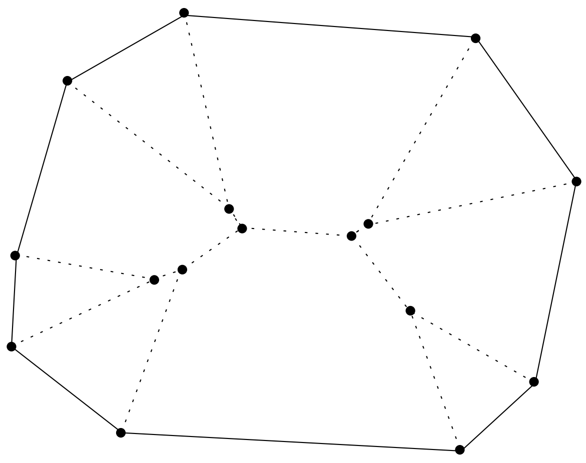

Figure 3.8: A simple polygon and its medial axis.

where a face is a set of points which are closest to the same site. Given a simple polygon, it is possible to partition it into a set of connected faces one for each vertex and edge, such that a face $F(s)$ adjacent to the vertex or (open) egde s is the set of all points inside the simple polygon that are closest to s. This partition of the simple polygon is identical to its Voronoi diagram where the open edges and vertices of the polygon are taken to be the sites. The medial axis of a simple polygon P is defined as the locus of the centers of all circles contained in P, such that the circles touch at least two edges of P. Thus the medial axis is essentially the same as (in fact, a subset of) the Voronoi diagram of the vertices and edges, as just defined. It has been proved by F. Chin et al. in [6], that the medial axis of a simple polygon can be computed in linear-time.

In the rest of this section we shall show that it is possible to use the medial axis of the simple polygons inside the bounding box B to find $\partial(M_{\leq n})$, with considerable gain in the efficiency of the algorithm. Clearly the Minkowski sum $\partial(M_{\leq n})$, lies completely inside B. Consider another bounding box B' which has its center at the same point as the smallest bounding box and whose width and height are both increased by an additional $2R$ as shown in Figure 3.7. Clearly $\partial(M_{\leq n})$ lies completely inside B', because of the definition of Minkowski sum. Assume that the sub-river $\mathcal{R}$ along with its bounding box B define two simple polygons P_1 and P_2. Let P_1^* be the partition of P_1 induced by its medial axis. If s is an edge of P_1 then let $F(s)$ be the face of P_1^* adjacent to s. Without loss of generality we will look into the construction of $P_1 \cap \partial(M_{\leq n})$, the construction of $P_2 \cap \partial(M_{\leq n})$ will be similar. Additionally notice that all those faces of P_1^* that are adjacent to an edge of B will not intersect with $\partial(M_{\leq n})$. Therefore we have:

Observation 3.4.1 *It is sufficient to consider only those faces of P_1^* that are adjacent to a segment of $\mathcal{R}$ for computing $P_1 \cap \partial(M_{\leq n})$.*

The general approach of the algorithm is to consider the intersection of the Minkowski sum with each of the faces of the partition P_1^*. For each segment $s_i \in \mathcal{R}$, the algorithm

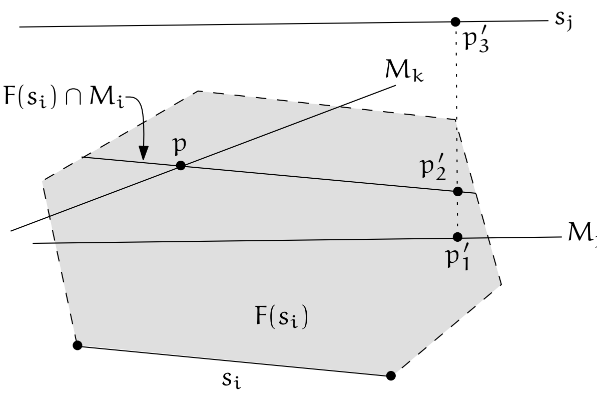

Figure 3.9: A segment of $\mathcal{R}$ and its Voronoi face.

computes $F(s_i) \cap \partial(M_{\leq n})$. Once all the faces of P_1^* are processed in this manner we obtain $P_1 \cap \partial(M_{\leq n})$. The reasons why the computation of $P_1 \cap \partial(M_{\leq n})$ is straightforward, given the medial axis of P_1 is because of the following properties:

- For some $s_i \in \mathcal{R}$ if $F(s_i) \cap \partial(M_i) = \emptyset$, then the face $F(s_i)$ does not contribute any edge to the output, which is $P_1 \cap \partial(M_{\leq n})$. This follows directly from the fact that in this case $F(s_i)$ lies completely inside M_i, so none of the edges of $M_{\leq n}$ inside $F(s_i)$ will be part of $\partial(M_{\leq n})$.

- For some $s_i \in \mathcal{R}$ if $F(s_i) \cap \partial(M_i) \neq \emptyset$, then the arc(s) $F(s_i) \cap \partial(M_i)$ will form a part of $P_1 \cap \partial(M_{\leq n})$. To see that this is in fact the case, consider the scenario in Figure 3.9. There are two cases to consider, depending on whether the boundary of another cigar intersects the boundary of M_i or not. Both cases are illustrated in Figure 3.9. A segment $s_i \in \mathcal{R}$ is shown along with its face $F(s_i)$ and intersections with some arbitrary cigars M_j and M_k. The claim is that only $F(s_i) \cap \partial(M_i)$ is part of $P_1 \cap \partial(M_{\leq n})$ inside $F(s_i)$ and not $F(s_i) \cap \partial(M_j)$ or $F(s_i) \cap \partial(M_k)$.

  - To see that $F(s_i) \cap \partial(M_j)$ is not part of $P_1 \cap \partial(M_{\leq n})$, consider $F(s_i) \cap \partial(M_j)$ in Figure 3.9. If s_j is below M_j in the diagram shown, then $F(s_i) \cap \partial(M_j)$ lies completely inside M_i, as a result $F(s_i) \cap \partial(M_j)$ will not be part of $P_1 \cap \partial(M_{\leq n})$. If s_j is above M_j in the diagram shown, then we select an arbitrary point p'_1 on $F(s_i) \cap \partial(M_j)$ such that an orthogonal line from p'_1 intersects $F(s_i) \cap \partial(M_i)$ and s_j at p'_2 and p'_3 respectively. Clearly $|\overline{p'_1 p'_3}| = R$ and $|\overline{p'_2 p'_3}| < R$. This implies that p'_2 is closer to s_j than to s_i, which violates the medial axis property. Therefore $F(s_i) \cap \partial(M_j)$ cannot lie between $F(s_i) \cap \partial(M_i)$ and s_i.

    In Figure 3.9, if $F(s_i) \cap \partial(M_j)$ were to lie above $F(s_i) \cap \partial(M_i)$ and in the interior of $F(s_i)$, then it again easy to see that this is impossible. If we assume this scenario, then for some point p on $F(s_i) \cap \partial(M_j)$, the shortest distance to s_j would be R. This will require that the shortest distance from p to s_i be less than R since p lies in the interior of $F(s_i)$. But the shortest distance from p to

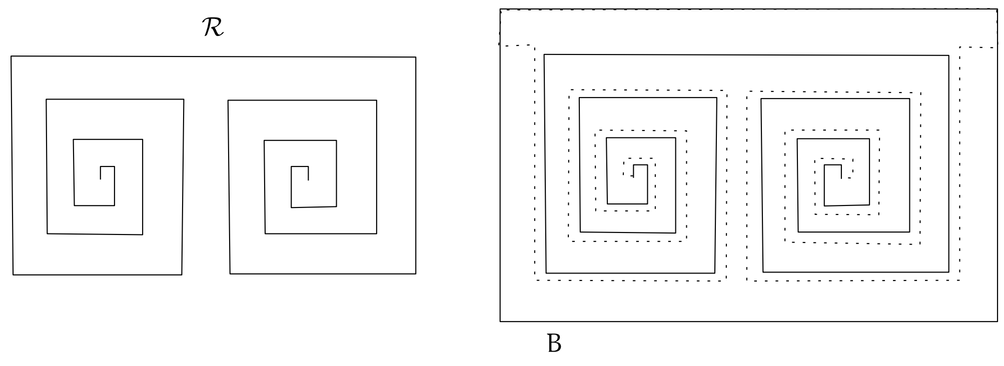

Figure 3.10: A sub-river with 'hidden' end-vertices.

s_i is in fact greater than R as it is above $F(s_i) \cap \partial(M_i)$. Therefore $F(s_i) \cap \partial(M_j)$ cannot lie above $F(s_i) \cap \partial(M_i)$ and in the interior of $F(s_i)$.

– To see that $F(s_i) \cap \partial(M_k)$ is not part of $P_1 \cap \partial(M_{\leq n})$, assume $F(s_i) \cap \partial(M_k)$ intersects $F(s_i) \cap \partial(M_i)$ at a point p in the interior of $F(s_i)$. From the definition of Minkowski sum, the shortest distance to s_k or s_i from p must be R. From the definition of medial axis of a simple polygon, this point p must be the center of a circle of radius R touching s_i and s_k, therefore it must lie *on* the medial axis. This contradicts our assumption that p lies in the interior of $F(s_i)$, therefore p *must* lie on the boundary of $F(s_i)$. Therefore $F(s_i) \cap \partial(M_k)$ is not part of $P_1 \cap \partial(M_{\leq n})$.

From these observations we conclude that it is sufficient to just add $F(s_i) \cap \partial(M_i)$ to $P_1 \cap \partial(M_{\leq n})$, and the rest of the arcs of $F(s_j) \cap \partial(M_{\leq n})$ can be ignored while processing the face $F(s_i)$. This observation makes the computation of $P_1 \cap \partial(M_{\leq n})$ highly efficient.

3.4.1 Partitioning into Simple Polygons

In some cases the partitioning of the sub-river into two simple polygons, with the help of the bounding box B can be accomplished easily by adding a single edges to the vertices v_0 and v_n connecting them to B. This is clearly only possible if it is possible to draw a straight line from v_0 and v_n to any of the edges of B. There can be many situations when this is not possible. Adding a single edge becomes impossible when one the end vertices of the sub-river is 'hidden' from the bounding box B. One such sub-river is shown in Figure 3.10. In this case where both v_0 and v_n are hidden, we draw a new sub-river $\mathcal{R}'$ consisting of two parts $\mathcal{R}'_1$ and $\mathcal{R}'_2$, very close to $\mathcal{R}$. $\mathcal{R}'_1$ joins v_0 to B and $\mathcal{R}'_2$ joins v_n with B as shown in the diagram. With this construction we get two simple polygons.

3.4.2 The Algorithm

Algorithm *SubRiverBufferZoneMedialAxis*($\mathcal{R}$, R)

```
Output: $M$ - the DCEL containing the edges and vertices of $\partial(M_{\leq n})$
1.  Compute the bounding boxes $B'$ and $B$.
2.  Partition $B$ and $\mathcal{R}$ into two simple polygons $P_1$ and $P_2$.
3.  Compute the medial axes of $P_1$ and $P_2$ using algorithm from [6].
4.  for $i \leftarrow 1$ to $n$
5.      do Compute $M_i$.
6.  for $P_1$ and $P_2$
7.      do for each face $f$ in the partition of $P_k$
8.             do if $f$ is adjacent to a segment $s_i$ of $\mathcal{R}$
9.                   then Compute the intersection $f \cap \partial(M_i)$ and add it to $M$, i.e.
                          $M \leftarrow M \cup (f \cap \partial(M_i))$.
10. return $M$.
```

Theorem 3.4.2 *Algorithm SubRiverBufferZoneMedialAxis computes* $\partial(M_{\leq n})$ *in linear time.*

Proof. The computation of the bounding boxes takes $O(n)$ time. Finding the two simple polygons takes $O(n)$ time since there are only $O(n)$ segments in $\mathcal{R} \cup B$. The computation of the partition and medial axes based on the algorithm in [6] also takes $O(n)$ time. Since there are only $O(n)$ faces in $P_1 \cup P_2$, the line 9 in the algorithm executes at most $O(n)$ times. Also computing the intersection of a face f with the cigar M_i is a constant time operation. Therefore we conclude that Algorithm *SubRiverBufferZone-MedialAxis* takes $O(n)$ time to find $\partial(M_{\leq n})$. $\square$

Chapter 4

The Three-Dimensional Case

4.1 Terminology and Notation

A three-dimensional polyhedral terrain can be defined as the graph of piecewise linear continuous function $f : S \mapsto \Re$, where $S \subset \Re^2$ is the domain of the terrain. We assume S to be a convex polygon. Similar to the two-dimensional case we will assume the terrain to extend to $z = -\infty$. The terrain shall be denoted by $\mathcal{T}$ abusing the notation slightly. The polygonal faces of the polyhedral terrain are assumed to be triangulated except the vertical *walls* that bound it. So the upper boundary of the terrain $\mathcal{T}$ in question is composed of n triangles $t_1, t_2, \ldots, t_n$. There are $O(n)$ vertices in the terrain which we label as v_{i1}, v_{i2} and v_{i3} for a given triangle t_i for $1 \le i \le n$. We shall denote the orthogonal projection of a region $r \subset \Re^3$ on the xy-plane by r^*. The orthogonal projection of the triangle t_i on the xy-plane is a triangle t_i^*. If we were to join the vertices of t_i to the corresponding vertices of t_i^* and extend these separate segments to infinity in both directions then the three lines define a unique infinite triangular prism $\Delta(t_i)$. More formally $\Delta(t_i) = \{(x, y, z_1) | (x, y, z_2) \in t_i \text{ and } z_1 \in (-\infty, +\infty)\}$. The triangle t_i along with the portion of $\Delta(t_i)$ that lies below t_i forms a semi-infinite prism, which will be denoted $\bar{t_i}$. So $\bar{t_i} = \{(x, y, z_1) | (x, y, z) \in t_i \text{ and } z_1 \le z\}$. For convenience each $\bar{t_i}$ will be called a semi-prism from now on. The terrain $\mathcal{T}$ is the union of all the semi-prisms, $\mathcal{T} = \bigcup_i \bar{t_i}$. To denote a subset of the semi-prisms, we shall use the convenient notation of -

$$\bar{t}_{\le a} = \bigcup_{1 \le i \le a} \bar{t}_i \ ,$$

$$\text{so we have: } \bar{t}_{\le n} = \bigcup_{1 \le i \le n} \bar{t}_i = \mathcal{T}.$$

$\mathcal{B}$ represents a ball of radius R centered at the origin. The Minkowski sum of $\mathcal{B}$ with the triangle t_i is denoted by k_i, that is $k_i = t_i \oplus \mathcal{B}$. The Minkowski sum of $\mathcal{B}$ with the semi-prism $\bar{t}_i$ is denoted by K_i, that is $K_i = \bar{t}_i \oplus \mathcal{B}$. Similar to the two-dimensional case we use $\partial(o)$ to denote the boundary of the object $o \subset \Re^3$. $\partial_u(o)$ is the upper part of $\partial(o)$ and $\partial_d(o)$ is the lower part of $\partial(o)$.

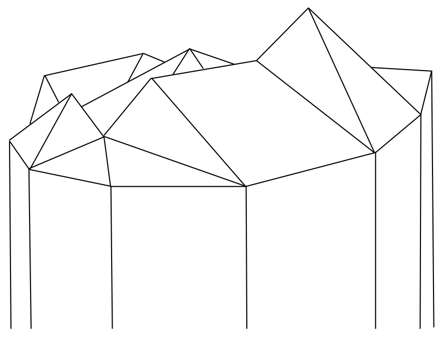

Figure 4.1: A polyhedral terrain

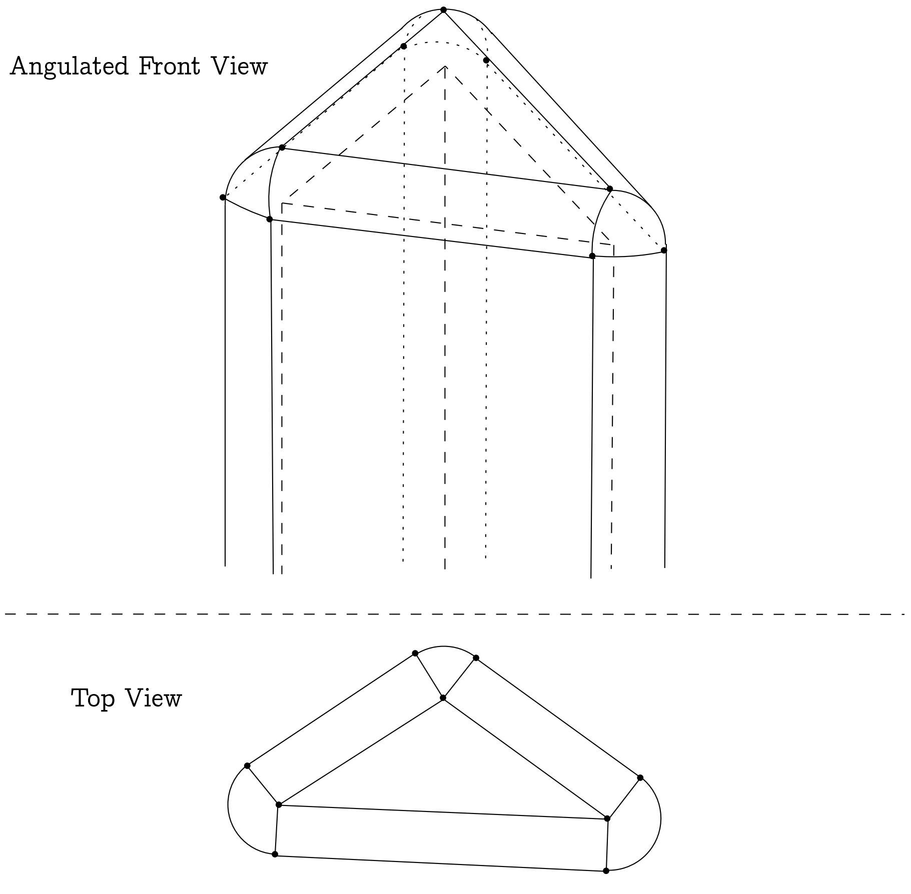

Figure 4.2: A semi-krepl

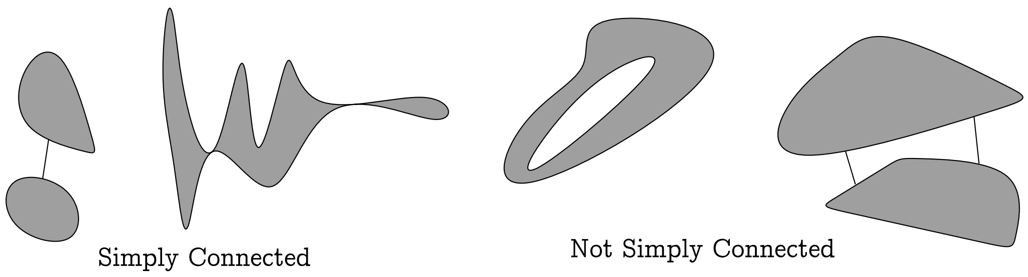

Figure 4.3: Simply connected sets

4.2 Preliminaries

Before we explore the properties of the Minkowski sum of the terrain with a ball, some definitions that are useful will be listed here. A *(polygonal) chain* $\mathcal{C} = (u_1, u_2, \ldots, u_n)$ is a planar straight line graph with the vertex set $\{u_1, u_2, \ldots, u_n\}$ and edge set $\{(u_i, u_{i+1}) | 1 \leq i < n\}$. The chain $\mathcal{C}$ is called a monotone chain with respect to a line l if the intersection of any line that is perpendicular to l with $\mathcal{C}$ is at most one point.

A polygon that is formed by a single closed polygonal chain, which does not intersect itself is called a *simple* polygon, that is, it has no holes and has non-intersecting edges. A simple polygon is said to be *monotone* if its boundary can be partitioned into two polygonal chains that are monotone with respect to the same line l. A set is called *simply connected*, if it is connected and every loop in the set can be contracted to a point. In other words the set is connected and does not have any holes. Similar to the definition of a *polygonally connected* set [5], we shall define a set M to be Ω-connected in $\Re^3$, if for any two points a and b in M there is a continuous function $f : [0,1] \mapsto M$ such that :

- $f(0) = a$ and $f(1) = b$.
- For distinct points $x, y \in (0,1), f(x) \neq f(y)$.
- The image of $[0,1]$ under f is the union of a finite number of line segments and arcs of constant description complexity.

Central to our analysis of the three-dimensional terrain is the assignment of an order to the semi-prisms. This is achieved in the same way as it is done by Asano et al, in [5].

Lemma 4.2.1 *[5] It is possible to define an ordering of the semi-prisms $\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_n$, denoted by π, such that the projection on xy-plane of $\bigcup_{i=1}^{j} \bar{t}_i$, for $1 \leq j \leq n$, is a monotone polygon with respect to fixed line l.*

This ordering can be found in $O(n)$ time. The notation $\bar{t}_{\leq a}$ will now have the definition of -

$$\bar{t}_{\leq a} = \bigcup_{1 \leq i \leq a} \bar{t}_{\pi(i)} \ ,$$

$$\text{so we have: } \bar{t}_{\leq n} = \bigcup_{1 \leq i \leq n} \bar{t}_{\pi(i)} = \mathcal{T}.$$

Additionally we shall assume that the semi-prisms are renumbered according to one such permutation π. So the semi-prism $\bar{t}_{\pi(i)}$ will simply be denoted by $\bar{t}_i$ from now on. Continuing, we consider the Minkowski sum of a single triangle with the sphere and explore some of its properties.

4.2.1 A Single Triangle

Consider the Minkowski sum of a single triangle with the sphere. It is composed of a triangular prism whose cross section at the center is the triangle itself and whose height is $2R$ where R is the radius of the sphere. There is a semi-cylinder whose axis is an edge of the triangle for each edge of the triangle. Finally at the three ends of this object there are parts of a sphere whose center is the vertex of the original triangle. We shall call the Minkowski sum of the triangle and the sphere a *krepl*, the plural form of *krepl* is *kreplach*, borrowing terminology from [3]. The krepl that is formed by the triangle t_i is denoted by k_i, that is $k_i = t_i \oplus \mathcal{B}$. The Minkowski sum of a 3D terrain and a sphere is the union of several such kreplach, where each krepl is positioned in a special manner with respect to other kreplach. The surface of any krepl k_i denoted as $\partial(k_i)$, can be partitioned into two surfaces, the upper part $\partial_u(k_i)$ and the lower part $\partial_d(k_i)$, so that any vertical ray from $y = +\infty$ intersects $\partial_u(k_i)$ before $\partial_d(k_i)$. For each k_i, the infinite volume that lies vertically below $\partial_u(k_i)$ will be called a semi-krepl K_i, as depicted in Figure 4.2. More formally $K_i = \{(x, y, z_1) | (x, y, z) \in \partial_u(k_i) \text{ and } z_1 \leq z\}$. The union of all the semi-kreplach is $\mathcal{K} = \bigcup_i K_i = \mathcal{T} \oplus \mathcal{B}$. We are interested in knowing the combinatorial complexity of $\partial(\mathcal{K})$ and an algorithm to compute it. The boundary of $\mathcal{K}$ is the boundary of the union of the individual K_i's,

$$\partial(\mathcal{K}) = \partial\left(\bigcup_i K_i\right).$$

The combinatorial complexity is defined as the number of faces, vertices and edges that form $\partial(\mathcal{K})$. Euler's formula for a connected planar embedded graph states that if such a graph has m_v vertices, m_e edges and m_f faces, then they are related by the expression $m_v - m_e + m_f = 2$. Since the projection of $\partial(\mathcal{K})$ on the xy-plane is a connected planar embedded graph, we can know that the number of faces and edges are linear with respect to the number of vertices. As a consequence of this, the focus will be on counting the number of vertices in $\partial(\mathcal{K})$ when the combinatorial complexity or the algorithm complexity has to be determined.

We already described that it is possible to define an ordering for the semi-prisms. The corresponding semi-kreplach of the semi-prisms will follow the same ordering. Like in two-dimensional discussion we will use a similar notation to denote a subset of the n semi-kreplach. For example,

$$K_{\leq a} = \bigcup_{1 \leq i \leq a} K_i\ ,$$

$$\text{so we have: } K_{\leq n} = \bigcup_{1\leq i\leq n} K_i = \mathcal{T} \oplus \mathcal{B} = \mathcal{K}.$$

Pseudosphere Property: Two solids a and b are said to satisfy the pseudosphere property if the solids $a\backslash b$ and $b\backslash a$ are topologically equivalent to a ball.

4.2.2 Properties of Kreplach

Directly from the definition of the Minkowski sum for a single triangle t_i and its corresponding krepl k_i we have :

Observation 4.2.2 *From every point of* t_i *the shortest distance to* k_i *is* R.

Since the terrain $\mathcal{T}$ is triangulated, for $i \neq j$, if the two triangles t_i and t_j are adjacent in $\mathcal{T}$ then their adjacency can be classified into two cases. *Edge-edge* if the whole of the edge of t_i coincides with the whole of the edge of t_j, or *vertex-vertex* if a vertex of t_i coincides with a vertex of t_j. For two triangles t_i and t_j which are edge-edge adjacent in $\mathcal{T}$, the union of their corresponding kreplach $\partial(k_i \cup k_j)$, will have a common cylindrical face of radius R and whose axis is the common edge. If the angle between the normals to the planes of the two triangles is more that $\pi/2$ then this common cylindrical face may form a face of $\partial(\mathcal{K})$ if no other face of $\partial(\mathcal{K})$ is directly above it. Similarly if these two triangles have a vertex-vertex adjacency, then $\partial(k_i \cup k_j)$ will have a common spherical face of radius R and whose center is the common vertex. Again if the angle between the normals to the planes of the two triangles is more that $\pi/2$ then this common spherical face may form a face of $\partial(\mathcal{K})$ if no other face of $\partial(\mathcal{K})$ is directly above it.

Observation 4.2.3 *For each vertex* v *in* $\partial_u(\mathcal{K})$ *there is a unique sphere of radius* R, $\mathcal{S}(v)$ *centered at* v *such that it touches at least 3 triangles of* $\mathcal{T}$.

Any vertex in $\partial_u(\mathcal{K})$ is formed by the intersection of the surfaces of at least three kreplach. Since this vertex lies on the surfaces of the kreplach, it lies at a distance of R from each of the corresponding triangles. Notice that like in the two-dimensional case the sphere can be in contact with more than 3 triangles of $\mathcal{T}$. Additionally, from the definition of a three-dimensional terrain we can deduce that :

Observation 4.2.4 *The interior of* $\Delta(t_i)$ *does not contain any vertex* v_j *of* $\mathcal{T}$.

The intersection of two non-adjacent kreplach results in a single closed curve or is empty. This result has already been proved by Agarwal and Sharir in [3].

Lemma 4.2.5 *[3] For* $i \neq j$, *if* t_i *and* t_j *are non-adjacent triangles of* $\mathcal{T}$, *then the intersection of the surfaces of any two kreplach* k_i *and* k_j; $\partial(k_i) \cap \partial(k_j)$, *is either empty or a single closed curve.*

Lemma 4.2.6 $\mathcal{B} \oplus \mathcal{T}$ *is a terrain*

Proof. The proof follows the same approach as in the two-dimensional case. It should be shown that the intersection of any vertical line with $\mathcal{B} \oplus \mathcal{T}$ is a ray infinite downwards. Therefore it is sufficient to show that for some point $\vec{p} = (x, y, z) \in \mathcal{T} \oplus \mathcal{B}$ all other points $\vec{p_d} = (x, y, z_d)$ such that $z_d \leq z$ also belong to $\mathcal{T} \oplus \mathcal{B}$.

Assume that point $\vec{p} = (x, y, z) \in \mathcal{T} \oplus \mathcal{B}$. There must be points $\vec{a} = (x_a, y_a, z_a) \in \mathcal{T}$ and $\vec{b} = (x_b, y_b, z_b) \in \mathcal{B}$ such that $\vec{a} + \vec{b} = \vec{p}$. This implies $x_a + x_b = x$, $y_a + y_b = y$ and $z_a + z_b = z$. Now assume that there is an arbitrary point $\vec{p_d} = (x, y, z_d)$ directly below $\vec{p}$ such that $z_d \leq z$. Choose a point $\vec{a_d}$ that is directly below $\vec{a}$. Since all points which are directly below $\vec{a}$ also belong to $\mathcal{T}$, $\vec{a_d}$ must also belong to $\mathcal{T}$ because of the terrain property. Let $\vec{a_d} = (x_a, y_a, z_d - z_b)$. Since $z_d \leq z$ it must hold that $z_d - z_b \leq z - z_b$, we already know that $z - z_b = z_a$ so $z_d - z_b \leq z - z_b = z_a$. This means that the choice of the z-coordinate of $\vec{a_d}$ is right and it is indeed directly below $\vec{a}$. To continue observe that

$$\vec{a_d} \in \mathcal{T} \text{ and } \vec{b} \in \mathcal{B} \text{ resulting in } \vec{a_d} \oplus \vec{b} = (x_a + x_b, y_a + y_b, z_d) = \vec{p_d} \in \mathcal{T} \oplus \mathcal{B}.$$

So we have shown that any arbitrary point $\vec{p_d}$ below $\vec{p}$ also belongs to $\mathcal{T} \oplus \mathcal{B}$. Therefore it can be concluded that $\mathcal{T} \oplus \mathcal{B}$ is a terrain.

Also notice that the projection of $\mathcal{B} \oplus \mathcal{T}$ coincides with the Minkowski sum of the projections of $\mathcal{B}$ and $\mathcal{T}$, that is, it is the Minkowski sum of a convex polygon and a circle, which is again convex.

The boundary of each semi-krepl is composed of spherical, cylindrical and planar surfaces. Therefore the boundary of the union of several semi-kreplach, will be composed of parts of the spherical, cylindrical and planar surfaces. □

4.3 A Lower-Bound Construction

The lower-bound construction consists of a central wedge with roughly $n/2$ faces. The wedge has its 'back' to the xz-plane, with $n/2$ grooves in the front. There are also $n/2$ identical needle-like triangular prisms with a constant number of faces. These prisms are placed very close to the wedge sides as shown in Figure 4.4. The step-wise construction is described below.

- To construct the central wedge first the construction of the projection of the wedge on the yz-plane is explained. We start with a square on the yz-plane with the end points $(0, 0, 0), (0, 0, R), (0, -R, R)$ and $(0, -R, 0)$, which forms the base of the wedge.

- To construct the grooves, a circle of radius equal to R is drawn on the yz-plane at the point $(0, 0, R)$ so that it is on the upper half plane and tangential to the y-axis. The upper right 90^0 segment of this circle from $(0, 0, 2R)$ to $(0, -R, R)$ is divided into $n/2$ strips by $n/2$ equidistant parallel lines as shown in Figure 4.4. In other words $(n/2) - 1$ equidistant points are marked on the line segment between $(0, 0, R)$

and $(0, 0, 2R)$, and from each of these points a line in the direction of negative y-axis is drawn till it meets the boundary of the circle. Vertical lines are drawn wherever the lines meet the boundary of the circle to the ray directly below it. The resulting construction is shown in the side view of the diagram.

- The boundary of the resulting steps-like polygonal shape along with the square on the yz-plane forms the cross section of the polyhedral central mountain. This shape is given considerable thickness in the direction of the positive x-axis.
- The $n/2$ needle-like triangular prisms are placed very close to the wedge, with a distance of at least $2R$ between them. The distances between the needles can also be determined by using the following method. The first needle is placed close to the wedge. A sphere of radius R is placed so that is touches the bottom-most groove of the wedge and the first needle. The sphere is then slid along this bottom-most groove till it sufficiently far away from the first needle. The second needle is then placed touching the other side of the sphere at this position. This is continued to get $n/2$ needles.

There are $\Omega(n)$ faces on the wedge and the triangular prisms. So the input combinatorial complexity of the terrain is $\Omega(n)$. A sphere of radius R corresponding to $\mathcal{S}(v)$ can be placed in one of these grooves so that it also is in contact with one of the prisms. This contact between three faces gives us a vertex of $\partial(\mathcal{K})$. Since we can also slide the sphere along the groove so that it touches all the prisms we get an additional $\Omega(n)$ contacts per groove. The same holds true for each grooves of the wedge. Since there are $n/2$ grooves, the total number of contacts is $\Omega(n^2)$. Hence the lower-bound for the Minkowski sum of $\mathcal{T}$ and a sphere is $\Omega(n^2)$.

4.4 The Upper-Bound

It directly follows from [3], that the combinatorial complexity of $\mathcal{T} \oplus \mathcal{B}$ is $O(n^{2+\epsilon})$ for any $\epsilon > 0$. It is interesting to note that three arbitrary kreplach need not necessarily satisfy the pseudosphere property. That is, for distinct i, j, k, if k_i, k_j, k_k are three kreplach of triangles of $\mathcal{T}$, then the two curves, $\partial(k_i) \cap \partial(k_j)$ and $\partial(k_i) \cap \partial(k_k)$, may intersect at more than 2 points. One such scenario is described in Figure 4.5. Notice that we can make t_j and t_k as thin as possible, long, and close to k_i, so that the corresponding curves $\partial(k_i) \cap \partial(k_j)$ and $\partial(k_i) \cap \partial(k_k)$, intersect at four points. This is a clear violation of the pseudosphere property, as a result we cannot conclude that the complexity of the union of n kreplach is $O(n^2)$ as we did in the two-dimensional case.

4.5 Algorithm

The algorithm for computing $\partial_u(K_{\leq n})$, will be an incremental construction algorithm like in the two-dimensional case. The general approach is similar to the algorithm presented

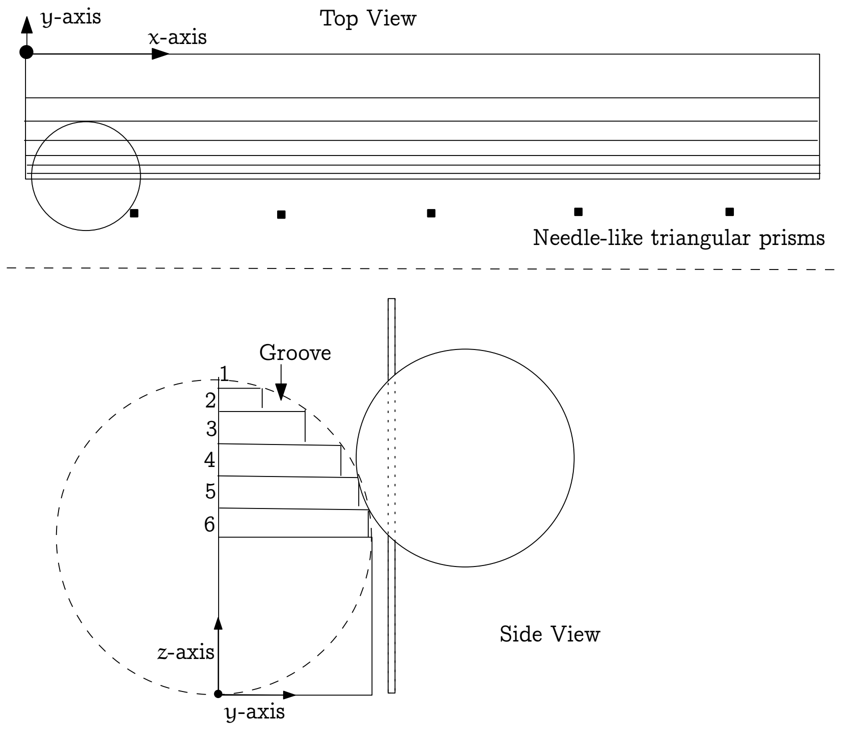

Figure 4.4: Top view and Side views of the lower bound construction.

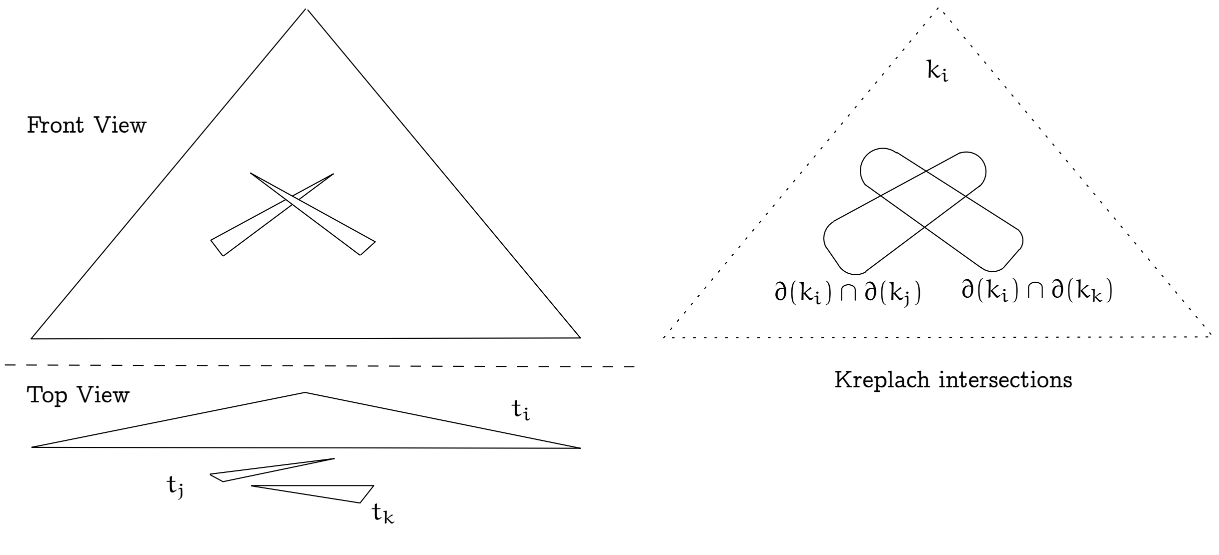
Front View
Top View
t_i
t_j
t_k
k_i
$\partial(k_i) \cap \partial(k_j)$
$\partial(k_i) \cap \partial(k_k)$
Kreplach intersections

Figure 4.5: Pseudosphere property violation scenario

in [5]. First a permutation π of the subscripts of the n triangles $t_1, t_2, \ldots, t_n$ is found, such that the projection of $\bigcup_{1\leq i\leq j} t_i$ on the xy-plane is a monotone polygon for $1 \leq j \leq n$. The triangles are renumbered according to this permutation so that t_i refers to $t_{\pi(i)}$ for $1 \leq i \leq n$. The Minkowski sum of a set A and a set $H = H_1 \cup H_2 \cup \ldots \cup H_m$, can be expressed as

$$A \oplus H = A \oplus \left(\bigcup_{1\leq i\leq m} H_i \right) = \bigcup_{1\leq i\leq m} (A \oplus H_i) = (A \oplus H_a) \cup \left(\bigcup_{i\neq a} A \oplus H_i \right).$$

Using the same idea for the terrain we get

$$\mathcal{B} \oplus \mathcal{T} = \mathcal{B} \oplus \left(\bigcup_{1\leq i\leq n} \bar{t}_i \right) = \bigcup_{1\leq i\leq n} (\mathcal{B} \oplus \bar{t}_i) = (\mathcal{B} \oplus \bar{t}_1) \cup \left(\bigcup_{2\leq i\leq n} \mathcal{B} \oplus \bar{t}_i \right).$$

This observation suggests computing $(\mathcal{B} \oplus \bar{t}_1) = K_1$ first. Subsequently K_2 is computed and their union $K_{\leq 2} = K_1 \cup K_2$ is found. In this manner the rest of the semi-kreplach $K_3, K_4, \ldots, K_n$ are added in the order defined by π. At any given point in the algorithm we will have a valid terrain $K_{\leq j}$, which is a subset of $K_{\leq n}$.

Algorithm *Mink3D*$(\mathcal{T}, R)$
Output: M the DCEL containing the edges, faces and vertices in $\partial_u(K_{\leq n})$

```
1.  Triangulate the polygonal faces of $\mathcal{T}$
2.  Determine the ordering $\pi$ so that the projection of $\bigcup_{1\leq i\leq j} t_i$ on the $xy$-plane is a
    monotone polygon for $1 \leq j \leq n$
3.  $M \leftarrow K_1$
4.  for $i \leftarrow 2$ to $n$
5.      do (* Update $M$, i.e. compute $M \leftarrow M \cup K_i$ *)
6.          ComputeUnion($\partial(M)$, $\partial(K_i)$)
7.  return $L$
```

The details of Algorithm *ComputeUnion* are explained later. First we shall look into the properties of the intersection $\partial(K_{<i}) \cap \partial(K_i)$ and prove that it is an Ω-connected arc.

4.5.1 Properties of $\partial(K_{<i}) \cap \partial(K_i)$

Lemma 4.5.1 $(\mathcal{B} \oplus \bar{t}_{\leq j})^* = \mathcal{B}^* \oplus (\bar{t}_{\leq j})^*$, *and therefore* $(\mathcal{B} \oplus \bar{t}_{\leq j})^*$ *is a monotone region without holes.*

Proof. First we shall prove the equality of the two sets. Consider *any two* points p and q, such that $p = (x_1, y_1, z_1) \in \mathcal{B}$ and $q = (x_2, y_2, z_2) \in \bar{t}_{\leq j}$. The Minkowski sum of these two points is $(x_1 + x_2, y_1 + y_2, z_1 + z_2) \in \mathcal{B} \oplus \bar{t}_{\leq j}$. The projection of this point must lie in the projection of the $\mathcal{B} \oplus \bar{t}_{\leq j}$, so we have $r = (x_1 + x_2, y_1 + y_2) \in (\mathcal{B} \oplus \bar{t}_{\leq j})^*$.

Since $p \in \mathcal{B}$ and $q \in \bar{t}_{\leq j}$, we have $p^* = (x_1, y_1) \in \mathcal{B}^*$ and $q^* = (x_2, y_2) \in (\bar{t}_{\leq j})^*$. So we have $p^* \oplus q^* = (x_1 + x_2, y_1 + y_2) = r \in \mathcal{B}^* \oplus (\bar{t}_{\leq j})^*$. This argument is applicable to all the pairs of points of $\mathcal{B}$ and $\bar{t}_{\leq j}$. This means for any two points $p \in \mathcal{B}$ and $q \in \bar{t}_{\leq j}$, the projection of their Minkowski sum is identical with the Minkowski sum of their projections. Therefore we can conclude that the two sets $(\mathcal{B} \oplus \bar{t}_{\leq j})^*$ and $\mathcal{B}^* \oplus (\bar{t}_{\leq j})^*$ are indeed equal.

$\mathcal{B}^*$ is a circle of radius R. The Minkowski sum of a circle $\mathcal{B}^*$ and a monotone polygon $(\bar{t}_{\leq j})^*$ is again monotone. Therefore $(\mathcal{B} \oplus \bar{t}_{\leq j})^* = K^*_{\leq j}$ is monotone region without holes. $\square$

Lemma 4.5.2 *If* $H_1, H_2 \subset \Re^2$ *and* H_1, H_2 *and* $H_1 \cup H_2$ *are simply connected then* $H_1 \cap H_2$ *is also simply connected.*

Proof. First we prove that $H_1 \cap H_2$ is connected and in the next step prove that it is *simply* connected. To prove connectedness, assume that $H_1 \cap H_2$ is composed of at least two disjoint sets. Let A and B be two of those sets. Let a and b be points in A and B respectively. Now since $a \in H_1$ and $b \in H_1$ there must be a curve $\alpha \subset H_1$ which joins both a and b as shown in Figure 4.6. Similarly there must be a curve $\beta \subset H_2$ which joins a and b. Consider the closed curve obtained by the union of α and β, this closed curve $\alpha \cup \beta$ clearly must be a part of $H_1 \cup H_2$, so $(\alpha \cup \beta) \subset H_1 \cup H_2$. We already know that $H_1 \cup H_2$ is simply connected, which implies that all points in the interior of $(\alpha \cup \beta)$ are also in $H_1 \cup H_2$. Let this region, which is bounded by the curves $(\alpha \cup \beta)$ be G. Let H_1' be the subset of H_1 that lies inside G, that is $H_1' = \{p | p \in H_1 \text{ and } p \in G\}$. Let H_2' be the subset of H_2 that lies inside G, that is $H_2' = \{p | p \in H_2 \text{ and } p \in G\}$. Since all points inside G belong to $H_1 \cup H_2$ we must have $H_1' \cup H_2' = G$ and $H_1' \cap H_2' \neq \emptyset$. If it were true that $H_1' \cap H_2' = \emptyset$ then there must be some point p which does not belong to H_1 or H_2, which will violate the simply connectedness of $H_1 \cup H_2$. To summarize:

$$H_1' \cup H_2' = G \tag{4.1}$$
$$H_1' \cap H_2' \neq \emptyset \tag{4.2}$$

Let A' and B' be the subsets of A and B respectively which lie inside G, that is $A' = \{p | p \in A \text{ and } p \in G\}$ and $B' = \{p | p \in B \text{ and } p \in G\}$. Clearly $A' \subset H_1'$, $A' \subset H_2'$ and $A' \subset (H_1' \cap H_2')$. Similarly $B' \subset H_1'$, $B' \subset H_2'$ and $B' \subset (H_1' \cap H_2')$. Thus we have:

$$(H_1' \cap H_2') \cap A' = A' \tag{4.3}$$
$$(H_1' \cap H_2') \cap B' = B' \tag{4.4}$$

If both $\partial(H_1) \cap G = \emptyset$ and $\partial(H_2) \cap G = \emptyset$ then $H_1' = H_2' = G$, which means that A' and B' are connected by a subset of $H_1 \cap H_2$, so we are done. If either $\partial(H_1) \cap G \neq \emptyset$ or $\partial(H_2) \cap G \neq \emptyset$ then let δ be that non-empty curve. Without loss of generality assume $\delta = \partial(H_1) \cap G$. Now if $H_1' \cap H_2'$ is connected then it follows that A' and B' are connected and we are done.

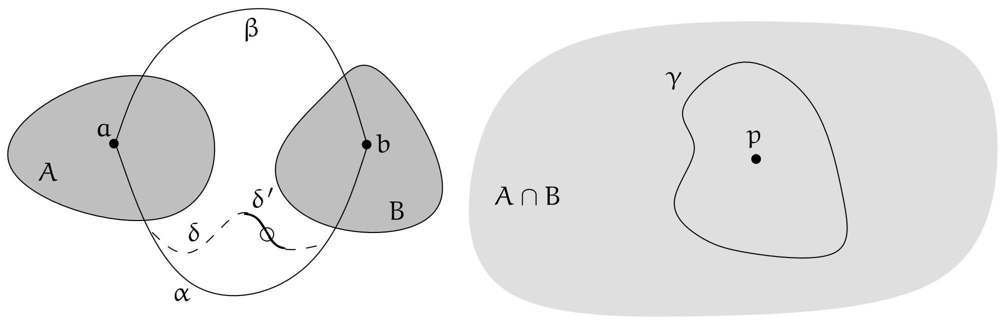

Figure 4.6: Illustration of proof of Lemma 4.5.2

To prove that $H'_1 \cap H'_2$ is also connected in all other cases we give a proof by contradiction. We will assume that $H'_1 \cap H'_2$ is not connected and subsequently show this leads to a contradiction. Assume that $H'_1 \cap H'_2$ is not connected, in this case $\delta \cap (H'_1 \cap H'_2)$ partitions δ into atleast three individual curves where two of the parts border $H'_1 \cap H'_2$. Atleast one of these parts of δ lies completely outside $H'_1 \cap H'_2$, let this part be δ'. On some arbitrary point of δ' draw a very small circle c. All points within this circle must belong to $H_1 \cup H_2$ since $H_1 \cup H_2$ is simply connected. Therefore all points in $c \cap H'_1$ belong to H'_1 and the rest of the points in c along with $c \cap \delta'$ must belong to H'_2. This is true for all points on δ'. Therefore all such points on δ' belong to both H'_1 and H'_2. This means that the curve $\delta \in (H'_1 \cap H'_2)$. This contradicts the assumption that $H'_1 \cap H'_2$ is disconnected. Therefore we conclude that $H_1 \cap H_2$ must be connected, which makes A' and B' connected.

To prove that $H_1 \cap H_2$ is *simply* connected consider a closed curve γ that lies completely inside $H_1 \cap H_2$, as shown in Figure 4.6. Let p be an arbitrary point inside γ. Since $\gamma \in H_1$ and H_1 is simply connected any point p inside γ must also belong to H_1. On the same lines, since $\gamma \in H_2$ and H_2 is simply connected any point p inside γ must also belong to H_2. Therefore all points inside γ are also present in $H_1 \cap H_2$. This arguments holds for any such curve $\gamma \in H_1 \cap H_2$, therefore it can be concluded that $H_1 \cap H_2$ is simply connected. □

Lemma 4.5.3 $(K_j \cap K_{<j})^* = K_j^* \cap K_{<j}^*$, *as a result* $(K_j \cap K_{<j})$ *is simply connected.*

Proof. In the first step of the proof it is shown that any point $(x, y) \in (K_j \cap K_{<j})^*$ also belongs to the set $K_j^* \cap K_{<j}^*$. Assume that a point $p = (x, y) \in (K_j \cap K_{<j})^*$. Since $(K_j \cap K_{<j})^*$ is the projection of the set $(K_j \cap K_{<j})$, there must be a z_1 such that $(x, y, z_1) \in (K_j \cap K_{<j})$. From this it follows that $(x, y, z_1) \in K_j$ and $(x, y, z_1) \in K_{<j}$. The projections of these individual points must belong to the projections of the individual sets, that is $(x, y, z_1)^* = (x, y) \in K_j^*$ and $(x, y, z_1)^* = (x, y) = K_{<j}^*$. So the point p lies in both the sets so we have $p = (x, y) \in K_j^* \cap K_{<j}^*$.

In the second step it is shown that any point $(x, y) \in K_j^* \cap K_{<j}^*$, also belongs to the set $(K_j \cap K_{<j})^*$, which will make the two sets equal. Assume that a point $p = (x, y) \in K_j^* \cap K_{<j}^*$.

This point must belong to both the sets so, $(x, y) \in K_j^*$ and $(x, y) \in K_{<j}^*$. Since these sets are projections, there must be a z_1 such that $(x, y, z_1) \in K_j^*$ and there must a z_2 such that $(x, y, z_2) \in K_{<j}^*$. Let $z_3 = \min(z_1, z_2)$. Since each K_i is vertically monotone and extends downward towards $-\infty$, the smallest z-coordinate must be present in both the sets. So we must have $(x, y, z_3) \in (K_j \cap K_{<j})$. Consequently $(x, y) \in (K_j \cap K_{<j})^*$. This proves the first part of the lemma.

Next we show that $K_j \cap K_{<j}$ is connected. We know that $(K_j \cap K_{<j})^* = K_j^* \cap K_{<j}^*$. That is, the projection of $K_j \cap K_{<j}$ on the xy-plane is the intersection of the convex projection of the semi-krepl K_j and the monotone region $K_{<j}^*$. To check if this region of intersection may have holes and therefore may not be a simply connected set, we observe that $K_j^* \cup K_{<j}^* = K_{\leq j}^*$. It has already been shown that $K_{\leq j}^*$ is a monotone region without holes from Lemma 4.5.1. Plugging in Lemma 4.5.2 we can conclude that $K_j^* \cap K_{<j}^*$ is simply connected. Since $K_j \cap K_{<j}$ is also vertically monotone, we conclude that $K_j \cap K_{<j}$ is simply connected. $\square$

Theorem 4.5.4 $\partial(K_j) \cap \partial(K_{<j})$ *is an Ω-connected arc.*

Proof. There are two steps to this proof. First it shall be shown that $K_j \cap \partial(K_{<j})$ and $\partial(K_j) \cap K_{<j}$ are Ω-connected surfaces. In the second step it will be shown that $\partial(K_j) \cap \partial(K_{<j})$ is an Ω-connected arc. To start, consider the surface $K_j \cap \partial(K_{<j})$, which lies completely inside K_j, assume that this surface is disconnected and is composed of two sets S_1 and S_2. Let $U_1 \subset \bar{t}_j$ and $U_2 \subset \bar{t}_j$ be the set of points in $\bar{t}_j$, that give rise to the sets S_1 and S_2 after the Minkowski sum of $\bar{t}_j$ and $\mathcal{B}$ is computed. In other words $U_1 = \{p \mid p \in \bar{t}_j \text{ and } (p \oplus \mathcal{B}) \cap S_1 \neq \emptyset\}$ and $U_2 = \{p \mid p \in \bar{t}_j \text{ and } (p \oplus \mathcal{B}) \cap S_2 \neq \emptyset\}$. Let u be a point in U_1 and v be a point in U_2.

From Lemma 4.5.1, we know that $(\mathcal{B} \oplus \bar{t}_{\leq j})^*$ is a monotone region without holes, and from Lemma 4.5.3, we know that $K_j \cap K_{<j}$ is connected. Moreover since the projection of $K_j \cap K_{<j}$ is simply connected and both K_j and $K_{<j}$ are terrains themselves, we conclude that $K_j \cap K_{<j}$ is *simply* connected, that is without holes. Since $K_j \cap K_{<j}$ is simply connected, it is possible to show that there is a connected set $S \subset \bar{t}_j$ which contains both u and v, with the special property that for any point $w \in S$, if we were to place $\mathcal{B}$ at w then it will always intersect $(K_j \cap K_{<j})$, so $(w \oplus \mathcal{B}) \cap (K_j \cap K_{<j}) \neq \emptyset$.

To see that this set S exists, imagine translating $\mathcal{B}$ to u first. Then translate $\mathcal{B}$ along a path inside $\bar{t}_j$, from u to v, so that at all times along this path, $\mathcal{B}$ intersects $(K_j \cap K_{<j})$. Naturally such a path *must exist*, since $u \oplus \mathcal{B}$ and $v \oplus \mathcal{B}$ both intersect $(K_j \cap K_{<j})$ by definition, and $(K_j \cap K_{<j})$ is connected. This path itself forms the set S. As a consequence of the existence of this set S, we notice that for any point $w \in S$, $w \notin \bar{t}_{<j}$, so $w \oplus \mathcal{B}$ will never lie completely inside $\partial(K_{<j})$, $w \oplus \mathcal{B}$ will always intersect $\partial(K_{<j})$. Since this happens for every point w_i of S, together, all the intersections of $w_i \oplus \mathcal{B}$ with $\partial(K_{<j})$ define a connected set in $K_j \cap \partial(K_{<j})$. This means that there is a 'bridge' between S_1 and S_2 making them connected. This contradicts the assumption that $K_j \cap \partial(K_{<j})$ is disconnected and composed of sets S_1 and S_2. Therefore $K_j \cap \partial(K_{<j})$ must be connected. Since the surface of each semi-krepl is already an Ω-connected surface, the surface of the union of several semi-kreplach must also necessarily be an Ω-connected surface.

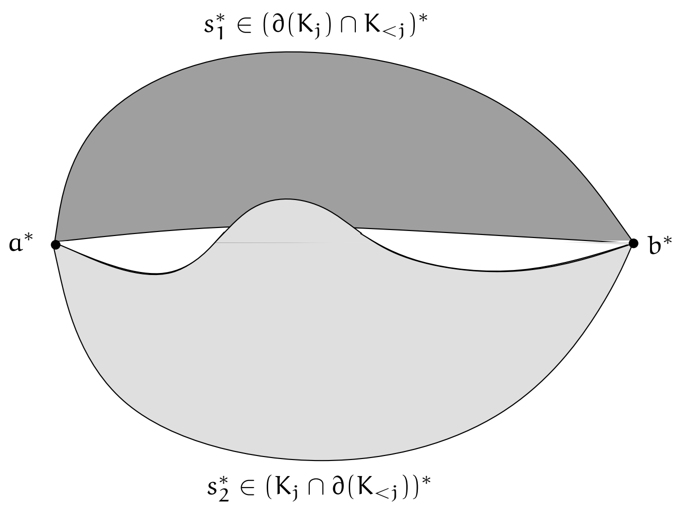
$s_1^* \in (\partial(K_j) \cap K_{<j})^*$
a^*
b^*
$s_2^* \in (K_j \cap \partial(K_{<j}))^*$

Figure 4.7: Projection of the curves s_1 and s_2 joining a and b

On similar lines it can be proved that $\partial(K_j) \cap K_{<j}$ is also an Ω-connected surface. Moving on to the second step of the proof, consider the intersection of the two surfaces $\partial(K_j) \cap \partial(K_{<j})$. Let a and b be two points such that $a, b \in \partial(K_j) \cap \partial(K_{<j})$. Each of these two points lie on the Ω-connected surfaces $K_j \cap \partial(K_{<j})$ and $\partial(K_j) \cap K_{<j}$. There must be Ω-connected arcs s_1 and s_2, lying on the surfaces $\partial(K_j) \cap K_{<j}$ and $K_j \cap \partial(K_{<j})$ respectively, each connecting a and b. Consider the region $R \subset \mathfrak{R}^3$ formed by the surfaces $\partial(K_j) \cap K_{<j}$ and $K_j \cap \partial(K_{<j})$ limited by s_1 and s_2. If a and b are not Ω-connected then R is not Ω-connected, which would mean that $K_j \cap K_{<j}$ is not simply-connected, which is not the case. To make this more clear, consider an impossible scenario in which $\partial(K_j) \cap \partial(K_{<j})$ is not an Ω-connected arc, this is illustrated Figure 4.7. In this *conceptual* figure the orthogonal projection of the region R along with the projections of s_1, s_2, a and b are shown. In the diagram the curves on the surfaces $K_j \cap \partial(K_{<j})$ and $\partial(K_j) \cap K_{<j}$ joining a and b do not coincide with each other. Any such scenario would mean that $K_j \cap K_{<j}$ is not simply-connected.

Secondly, the degree of each vertex of $\partial(K_j) \cap \partial(K_{<j})$ must be two, otherwise with a slight perturbation of $\mathcal{T}$ a curve $\partial(K_j) \cap \partial(K_{<j})$ which is disconnected is obtained.Thus we conclude that $\partial(K_j) \cap \partial(K_{<j})$ is an Ω-connected arc. □

When two triangles t_i and t_j have a common edge, then the union of the corresponding semi-kreplach have a common cylindrical face. This may seem contrary to the result of Theorem 4.5.4. This can be resolved as follows. The common cylindrical face can be assumed to be belonging to one of the two semi-kreplach and only one of the common edges is assumed to be the actual intersection. Similar assumption is done for adjacent semi-kreplach with common spherical faces. This way intersection of adjacent semi-kreplach with common faces are assumed to result in Ω-connected arcs instead of Ω-connected faces.

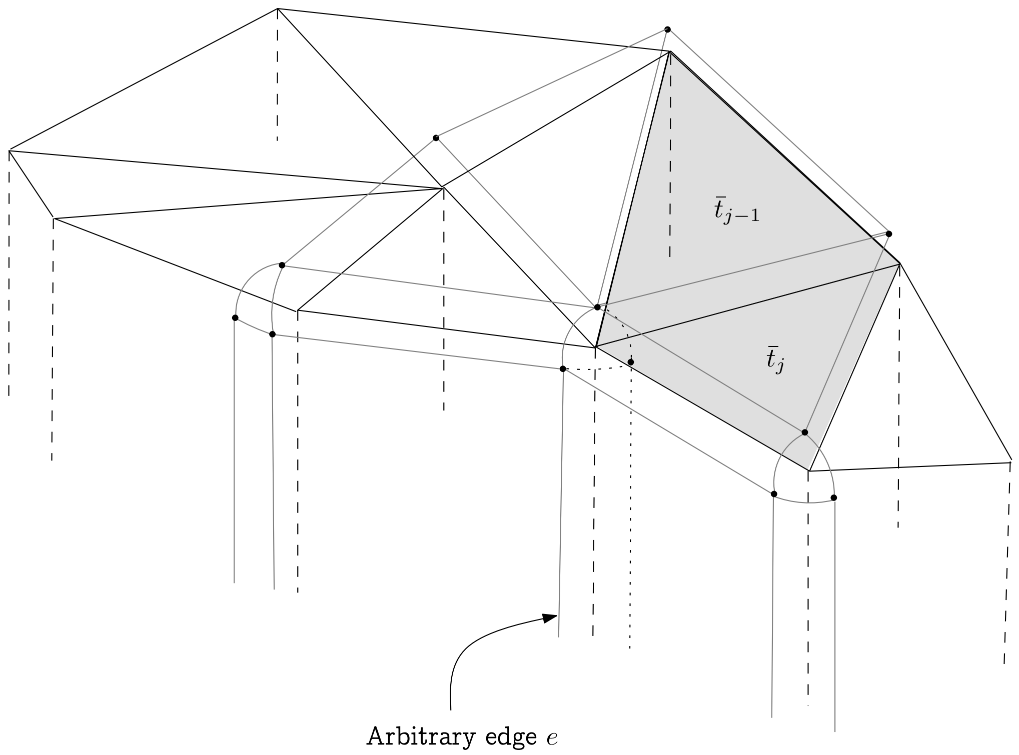

Figure 4.8: Finding an arbitrary edge of the union $\partial(\mathsf{K}_\mathsf{j}) \cup \partial(\mathsf{K}_{<\mathsf{j}})$

4.5.2 Computing the Union $\partial(\mathsf{K}_\mathsf{j}) \cup \partial(\mathsf{K}_{<\mathsf{j}})$

We will now describe the details of Algorithm *Mink3D*. The method of finding the union $\partial(\mathsf{K}_\mathsf{j}) \cup \partial(\mathsf{K}_{<\mathsf{j}})$ should be efficient for a good algorithm. The method described here is similar to the method described in [5]. Let γ denote the Ω-connected arc $\partial(\mathsf{K}_\mathsf{j}) \cap \partial(\mathsf{K}_{<\mathsf{j}})$. Let $\mathcal{F}_\mathsf{j}$ be the set of faces of $\partial(\mathsf{K}_\mathsf{j})$ that contribute some face to γ. Similarly let $\mathcal{F}_{<\mathsf{j}}$ be the set of faces of $\partial(\mathsf{K}_{<\mathsf{j}})$ that contribute some face to γ.

Since there is at least one and at most two overlapping edges, of the triangle t_j and the surface $\mathsf{t}_{<\mathsf{j}}$, we can find an arbitrary edge e of γ. The edge e is formed by the intersection of one face in $\partial(\mathsf{K}_\mathsf{j})$ and the other face in $\partial(\mathsf{K}_{<\mathsf{j}})$. This face can be found in constant time. The reason for this is that we have already defined the ordering π which will ensure that there are no semi-kreplach on at least one side of K_j, so that we always have a vertical face of $\partial(\mathsf{K}_\mathsf{j})$ that will intersect with a vertical face of $\partial(\mathsf{K}_{\mathsf{j}-1})$. This is illustrated in Figure 4.8.

Subsequently faces which are adjacent to the current faces and which contribute an edge to γ, are checked for intersection. The faces determined in this fashion form the sets $\mathcal{F}_\mathsf{j}$ and $\mathcal{F}_{<\mathsf{j}}$. The entire curve γ is found in this manner. For any face belonging to $\mathcal{F}_\mathsf{j}$ or $\mathcal{F}_{<\mathsf{j}}$, either the entire face or a part of it will be part of the surface $\partial(\mathsf{K}_\mathsf{j}) \cup \partial(\mathsf{K}_{<\mathsf{j}})$, only this visible part of the face has to be retained to compute the union and the hidden portions have to be deleted. These portions of each face of $\mathcal{F}_j$ and $\mathcal{F}_{<j}$ that are to be retained are computed next. The retained portions of each face of $\mathcal{F}_j$ and $\mathcal{F}_{<j}$ are then 'attached' to γ to get the structure γ'. Finally to compute the union, the faces of $\mathcal{F}_j$ and $\mathcal{F}_{<j}$ are removed from $\partial(K_j)$ and $\partial(K_{<j})$ respectively and the remaining portions of $\partial(K_j)$ and $\partial(K_{<j})$ along with the newly created portion γ' are joined together. This is described in the Algorithm *ComputeUnion*.

```
Algorithm ComputeUnion($\partial(K_j), \partial(K_{<j})$)
Output: DCEL containing the edges, faces and vertices of $\partial(K_{\leq i})$
1.  Find a single arbitrary edge $e$ of $\gamma = \partial(K_j) \cap \partial(K_{<j})$ that is formed by the intersection
    of vertical faces of $\partial(K_j)$ and $\partial(K_{j-1})$.
2.  Let $f_1 \in \partial(K_{<j})$ and $f_2 \in \partial(K_j)$ be the faces whose intersection is the edge $e$. Initialize
    the sets $\mathcal{F}_{<j} \leftarrow \{f_1\}$, $\mathcal{F}_j \leftarrow \{f_2\}$. Mark $f_1$ and $f_2$ as 'unprocessed'.
3.  If the face $f \in \mathcal{F}_{<j}$ is 'unprocessed', then check all 'unprocessed' adjacent faces of $f$
    for intersection with $\partial(K_j)$. If an adjacent face is completely inside $\partial(K_j)$ then mark
    it for deletion later on, also mark it as 'processed'. If an adjacent face intersects
    $\partial(K_j)$ then add it to $\mathcal{F}_{<j}$ and mark it as 'unprocessed'. Mark $f$ as being 'processed'.
    Continue till all faces in $\mathcal{F}_{<j}$ are marked as 'processed'.
4.  If the face $f \in \mathcal{F}_j$ is 'unprocessed', then check all 'unprocessed' adjacent faces of $f$ for
    intersection with $\partial(K_{<j})$. If an adjacent face is completely inside $\partial(K_{<j})$ then mark
    it for deletion later on, also mark it as 'processed'. If an adjacent face intersects
    $\partial(K_{<j})$ then add it to $\mathcal{F}_j$ and mark it as 'unprocessed'. Mark $f$ as being 'processed'.
    Continue till all faces in $\mathcal{F}_j$ are marked as 'processed'.
5.  Starting from the edge $e \in f_1 \in \partial(K_{<j})$, walk around the edges of the faces of $\mathcal{F}_{<j}$
    determining intersection with faces of $\mathcal{F}_j$ to compute $\gamma = \partial(K_j) \cap \partial(K_{<j})$ completely.
    This walk also helps in computing the visible portions of the faces of $\mathcal{F}_{<j}$ and $\mathcal{F}_j$.
6.  Compute $\gamma'$ as follows: For each face $f \in \mathcal{F}_j$, the visible portions of $f$ that must be
    retained are linked with $\gamma$. Similarly for each face $f \in \mathcal{F}_{<j}$, the visible portions of $f$
    that must be retained are linked with $\gamma$. The resultant structure is $\gamma'$.
7.  Remove from $\partial(K_j)$ and $\partial(K_{<j})$: 1) those vertices that are inside $\partial(K_j) \cup \partial(K_{<j})$ hence
    invisible 2) those edges which are incident on invisible vertices and those intersecting
    $\gamma$.
8.  Remove all the invisible faces of $\partial(K_j)$ and $\partial(K_{<j})$ that were previously determined,
    that is those that lie completely in the interior of $\partial(K_j) \cup \partial(K_{<j})$.
9.  Remove all the faces of $\mathcal{F}_j$ from $\partial(K_j)$ and all the faces of $\mathcal{F}_{<j}$ from $\partial(K_{<j})$.
10. The remaining structure of $\partial(K_j)$, $\partial(K_{<j})$ and $\gamma'$ are linked together.
11. return $\partial(K_{\leq i})$.
```

Lemma 4.5.5 *If v_j is the number of vertices that are deleted from $\partial(K_j)$ and $\partial(K_{<j})$ and $|\gamma|$ is the number of vertices in γ, then Algorithm-ComputeUnion takes $O(|\gamma| + v_j)$ time.*

Proof. Finding the arbitrary edge e in step-1, takes constant time as described previously. In step-3 the faces present in $\mathcal{F}_{<j}$ are found. This takes time proportional to the

number of faces in $\mathcal{F}_{<j}$, which is almost $|\gamma|$, plus the number of faces in $\partial(K_{<j})$ which have to be deleted, which is less than ν_j. In step-4 the faces present in $\mathcal{F}_j$ are found. This takes time proportional to the number of faces in $\mathcal{F}_j$, which is constant, plus the number of faces in $\partial(K_j)$ which have to be deleted, which is also a constant value. Step-5 computes γ and also the visible portions of the faces in $\mathcal{F}_{<j}$ and $\mathcal{F}_j$, this is the most time consuming step of the algorithm. Computing γ is clearly proportional to $|\gamma|$. Computing the visible portion of a single face is proportional to the number of other faces that intersect with it. Since in total there are $|\mathcal{F}_{<j}| + |\mathcal{F}_j|$ faces, the total number of such face-face intersections that have to be computed is proportional to $|\gamma|$. Computing γ' takes $O(|\gamma|)$ time.

Removing the invisible vertices and edges in step-7 takes time proportional to the total number of faces that have to be deleted which is ν_j. Step-8 takes $O(\nu_i)$ time. Step-9 takes time proportional to $|\mathcal{F}_{<j}| + |\mathcal{F}_j|$ which is $O(|\gamma|)$. Linking the remaining structures takes $O(|\gamma|)$ in step-10. Therefore Algorithm *ComputeUnion* takes $O(|\gamma| + \nu_j)$ time. □

Based on the fact that $\partial(K_j) \cap \partial(K_{<j})$ is an Ω-connected arc, it is possible to make further observations. It is our claim that for distinct i, j, k, the curves $\partial(K_i) \cap \partial(K_j)$ and $\partial(K_i) \cap \partial(K_k)$ on the surface $\partial(K_i)$ intersect in at most two points. The conclusive proof of this fact has been elusive till now. But if the claim were true then it is easy to show that the complexity of $\partial(K_{\leq n})$ is in fact $O(n\lambda_2(n))$, where $\lambda_2(n)$ is the Davenport-Schinzel sequence of order two.

Conjecture 4.5.6 *The combinatorial complexity of* $\partial(K_{\leq n})$ *is* $O(n\lambda_2(n))$.

Theorem 4.5.7 *Algorithm Mink3D computes the Minkowski sum in* $O(n^{2+\epsilon})$ *for any* $\epsilon > 0$.

Proof. The computation of a single semi-krepl can be done in constant time. The computation of all the n semi-krepl takes $O(n)$ time. The ordering of the semi-krepl can be found in $O(n)$ time as explained before. Computing the union of the over the n semi-kreplach takes the most time. From Lemma 4.5.5 we know that to compute union of $\partial(K_j) \cap \partial(K_{<j})$ it takes $O(|\gamma_j| + \nu_j)$, where $\gamma_j = \partial(K_j) \cap \partial(K_{<j})$. Therefore the total time is proportional to $\sum_{1 \leq i \leq n} |\gamma_i| + \nu_i$, which is the total number of vertices in $\partial(K_{\leq n})$ and the total number of vertices that are deleted. This is also the combinatorial complexity of the *arrangement* of the n semi-kreplach. Each infinite prism in $\mathcal{T}$ can be decomposed into 7 triangles by giving the prism some finite, large enough height. This will result in a total of $7n$ kreplach in the arrangement. From [3] it is clear that the combinatorial complexity of the arrangement of $7n$ kreplach is $O(n^{2+\epsilon})$ for any $\epsilon > 0$. Therefore the time complexity of Algorithm *Mink3D* is $O(n^{2+\epsilon})$ for any $\epsilon > 0$. □

Chapter 5

Conclusion

In this paper we have studied the problem of computing the Minkowski sum of piecewise linear functions in two and three-dimensions with a disk and ball respectively. In the two-dimensional scenario, for the two-dimensional terrain and a disk, we have shown the combinatorial complexity to be linear and have proved exact bounds for it. An algorithm that computes this Minkowski sum in linear time has been given. The problem of finding the Minkowski sum of a disk and a finite, piecewise-linear, non self intersecting curve was determined to have linear combinatorial complexity, an algorithm for computing this Minkowski sum has been provided that takes linear time.

In the three-dimensional case, detailed results and observations on the geometry of the Minkowski sum of a three-dimensional polyhedral terrain with a sphere have been established. An algorithm that computes this Minkowski sum in $O(n^{2+\epsilon})$, for any $\epsilon > 0$ has been shown. Additionally results describing the quadratic lower-bound of the complexity of the Minkowski sum and a conjecture on the upper-bound claiming $O(n\lambda_2(n))$, were also given.

The next step for this study would be to look into the proof of the conjecture on the upper-bound, in the three-dimensional case.

Bibliography

[1] Agarwal, P. K., Flato, E., and Halperin, D. Polygon decomposition for efficient construction of minkowski sums. In *ESA* (2000), M. Paterson, Ed., vol. 1879 of *Lecture Notes in Computer Science*, Springer, pp. 20–31.

[2] Agarwal, P. K., and Sharir, M. Motion planning of a ball amid segments in three dimensions. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms* (1999), pp. 21–30.

[3] Agarwal, P. K., and Sharir, M. Pipes, cigars, and kreplach: the union of minkowski sums in three dimensions. *Discrete & Computational Geometry 24* (2000), 645–657.

[4] Aronov, B., and Sharir, M. On translational motion planning of a convex polyhedron in 3-space. *SIAM Journal on Computing 26* (1997), 1785–1803.

[5] Asano, T., Hernández-Barrera, A., and Nandy, S. C. Translating a convex polyhedron over monotone polyhedra. *Comput. Geom 23* (2002), 257–269.

[6] Chin, F. Y. L., Snoeyink, J., and Wang, C. A. Finding the medial axis of a simple polygon in linear time. *Discrete & Computational Geometry 21* (1999), 405–420.

[7] de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. *Computational Geometry Algorithms and Applications.* Springer-Verlag, 1997.

[8] Dobkin, D., Hershberger, J., Kirkpatrick, D., and Suri, S. Implicitly searching convolutions and computing depth of collisiont. In *Algorithms: International Symposium (SIGAL '90)* (1990), pp. 165–180.

[9] Earthtrends. http://earthtrends.wri.org/.

[10] Guibas, L., Ramshaw, L., and Stolfi, J. A kinetic framework for computational geometry. In *24th Annual IEEE Symposium on Foundations of Computer Science* (1983), pp. 100–111.

[11] Guibas, L. J., and Seidel, R. Computing convolutions by reciprocal search. *Discrete & Computational Geometry 2* (1987), 175–193.

[12] Har-peled, S., Chan, T. M., Aronov, B., Halperin, D., and Snoeyink, J. The complexity of a single face of a minkowski sum. In *Proc. 7th Canad. Conf. Comput. Geom* (1995), pp. 91–96.

[13] Hernandez-Barrera. Computing the minkowski sum of monotone polygons. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems* (1997), 218–222.

[14] Kaul, A., O'Connor, M., and Srinivasan, V. Computing minkowski sums of regular polygons. In *Proc. 3rd Canad. Conf. Comput. Geom* (1991), pp. 74–77.

[15] Kedem, K., Livne, R., Pach, J., and Sharir, M. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete and Computational Geometry 1* (1986), 59–71.

[16] Li, Z., and Milenkovic, V. A compaction algorithm for non-convex polygons and its application. In *Proc. 9th ACM Symposium on Computational Geometry* (1993), pp. 153–162.