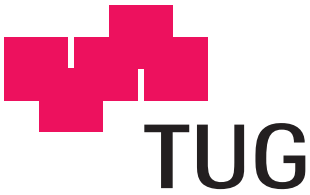
Dissertation

Algorithms for 3D Objects using Unions of Balls

Bernhard Kornberger

März 2010

Institut für Grundlagen der Informationsverarbeitung
Technische Universität Graz
Vorstand: o.Univ.-Prof.Dr.Wolfgang Maass



Gutachter und Betreuer: Univ.-Prof. DI. Dr. Franz Aurenhammer
Institut für Grundlagen der Informationsverarbeitung
Technische Universität Graz

Gutachter: Univ.-Prof. DI. Dr. Günter Rote
Institut für Informatik
Freie Universität Berlin

This thesis was written in partial fulfillment of the requirements for the degree of "Doktor der technischen Wissenschaften" at Graz University of Technology, Austria. It was carried out during the period of August 2005 to March 2010, in which both, research and development took place.

This work was supervised by Prof. DI. Dr. Franz Aurenhammer from the Institute for Theoretical Computer Science, Inffeldgasse 16b/1, 8010 Graz, Austria. My research was funded by the FWF Joint Research Project "Industrial Geometry", S9205-N12, at which I was employed as researcher until December 2009. Parts of this thesis have been published in scientific journals and conferences.

Abstract

Balls are among the simplest geometric objects, and they allow for computationally cheap operations like change of their size and inclusion tests. The structural properties of a union of balls can also be computed quickly, namely using their power diagram and its dual regular triangulation, respectively. This makes unions of balls an ideal support structure for approximation algorithms.

Our research partially relies on a known approach which uses the Voronoi diagram of a set of sample points of an object to compute a set of balls, whose union approximates this object. This approach yields unreasonably large, possibly unstable sets of balls. We have developed a technique to extract small, stable subsets, which efficiently approximate the object. This kind of representation allows for computation of stable medial axis approximations, which we substantiate by examples. Further, we have developed a new algorithm for the computation of Minkowski sums of such representations. We have compared this algorithm to existing exact and approximate solutions, and it scores well with respect to running time and robustness.

A completely different approach is the construction of a set of balls, whose union encloses the surface of an object on both sides like a thick skin. Thereby, the object is only given by an unorganized sample point cloud, and from the balls we can — optionally after pruning — reconstruct the object's surface.

This work has evolved in the rather academic area of computational geometry. We believe that scientific research has additional value, when the results find their way to real life. We bridge over with robust implementations of our algorithms.

Kurzfassung

Kugeln gehören zu den einfachsten geometrischen Objekten, und sie erlauben rechnerisch billige Operationen, wie etwa Größenänderungen und Inklusionstests. Die strukturellen Eigenschaften einer Vereinigung von Kugeln können ebenfalls schnell berechnet werden, nämlich über deren Power-Diagramm bzw. die dazu duale reguläre Triangulierung. Das macht Vereinigungen von Kugeln zu einer idealen Grundlage für Approximationsalgorithmen.

Diese Arbeit beruht teilweise auf einem bekannten Ansatz, der aus dem Voronoi-Diagramm einer Menge von Samplepunkten eines Objekts eine Kugelmenge generiert, deren Vereinigung dieses Objekt annähert. Dieser Ansatz erzeugt unverhältnismäßig große, möglicherweise instabile Kugelmengen. Daher haben wir ein Verfahren entwickelt, das daraus kleine, stabile Teilmengen extrahiert, die das Objekt effizient approximieren. Diese Art der Repräsentation erlaubt das Berechnen stabiler Mittelachsen-Approximationen, wofür wir Beispiele zeigen. Wir haben weiters ein neues Verfahren zur Berechnung von Minkowski-Summen solcher Repräsentationen entwickelt, das sich gegenüber bestehenden exakten und approximativen Lösungen als robust und schnell erweist.

Ein ganz anderer Ansatz ist die Konstruktion von Kugelmengen, deren Vereinigung die Oberfläche eines Objekts wie eine dicke Haut beidseitig umschließt. Dabei ist dieses Objekt nur durch eine lose Punktwolke gegeben, und aus der konstruierten Kugelmenge können wir – wahlweise nach Ausdünnen – die Oberfläche des Objekts rekonstruieren.

Diese Arbeit ist im eher akademischen Bereich der rechnerischen Geometrie entstanden. Wir glauben fest, daß wissenschaftliche Forschung den größten Nutzen hat, wenn die Resultate ihren Weg ins wirkliche Leben finden. Diese Verbindung stellen wir mit robusten Implementierungen unserer Algorithmen her.

Danksagung

Ich danke meiner Frau Birgit für die elf gemeinsamen Jahre, die besser nicht sein hätten können. Neben so vielen guten und wertvollen Dingen, die sie in mein Leben bringt, hat sie auch diese Arbeit korrekturgelesen. Danke, Birgit.

Zu sehr großem Dank bin ich meinem Betreuer, Franz Aurenhammer, verpflichtet. Er ist ein großer Wissenschafter, dem auch Humor und Gitarre nicht fehlen. Ich habe viel von ihm gelernt, und er hat mir so manchen wissenschaftlichen Schubs in die richtige Richtung versetzt. Ein großer Dank geht auch an die Kollegen, die ich immer in guter Erinnerung behalten werde, sei es für wertvolle Diskussionen oder für die guten gemeinsamen Abende. Und ich danke recht herzlich Wolfgang Slany für die Unterstützung, die er mir zukommen hat lassen. Ich weiß das sehr zu schätzen.

Ich danke meinen Co-Autoren Astrid Sturm und Günter Rote aus Berlin. Astrid und ich haben uns gemeinsam viele Stunden lang den Kopf zerbrochen und dabei so manches Problem geknackt. Aber wenn immer ein Problem härter war als unsere beiden Köpfe, war Günter zur Stelle, der uns dann oft erstaunlich schnell weiterhelfen konnte. An die Forschungsaufenthalte bei Euch in Berlin werde ich immer gerne zurückdenken.

Jedenfalls danken möchte ich auch meinen Forschungspartnern und Co-Autoren der Universität Innsbruck, der TU Wien und der JKU Linz für die Zusammenarbeit und für die gemeinsamen Aktivitäten. Ein großer Dank geht auch an die Geometrica Group in Sophia Antipolis, deren Gast ich mehrmals war. Ich habe dort wichtige Details über CGAL erfahren, und das war sehr hilfreich für meine Arbeit.

Doch nun wieder zurück zur Familie: Meine bisherige ist spätestens seit meiner Hochzeit um Birgit, Birgit's Familie und einige enge Freunde gewachsen. Ich weiß gar nicht, wo ich anfangen könnte. Ich danke Euch allen für Eure Unterstützung und für die guten Momente, die wir immer wieder miteinander erleben.

Graz, im März 2010 Bernhard Kornberger

Contents

Chapter 1

Introduction

In this chapter, we give an overview of the problems that have been considered. We shortly review existing work, and we state our contributions. The subsequent chapters will then cover the particular problems, the related work, and our contributions in detail.

1.1 Problem statement and contributions

1.1.1 Discrete medial axis (transforms)

For humans it is natural to recognize objects by features on their surface, by their dimensions or whatever is visible of the objects. In contrast, it is much more natural for algorithms to treat an object $\mathcal{O}$ by its skeletal structure, i.e., by its *medial axis* $\mathrm{M}(\mathcal{O})$, [DZ04, FLM03]. The medial axis has important applications in mesh generation, object simplification and analysis of an object. Related to this important geometric shape is the *medial axis transform*, which is an (infinite) set of balls, centered at $\mathrm{M}(\mathcal{O})$, whose union equals $\mathcal{O}$. A finite approximation, which we call the *discrete medial axis transform*, is particularly beneficial for collision detection, Minkowski sum computation, and approximate medial axis computation, to name only a few.

Groundbreaking work exists to compute a topologically correct discrete medial axis transform of an object $\mathcal{O}$, which is given by a sample point cloud S. But the sheer Voronoi approach, as described in [AB99, AK00], is very sensitive to undersampling, i.e., it works only correctly for r-sampled inputs (see Chapter 3 for details). A second problem is the high density of the output, which is due to the fact that almost every input sample point causes a ball in the resulting discrete medial axis transform. A third problem of the approach is that it (correctly) reproduces the unstable behavior of the real medial axis transform of $\mathcal{O}$, which leads to small, surface-near balls.

Our contribution includes a labeling method that reliably distinguishes inner from outer Voronoi vertices, when $\mathcal{O}$ is given by a triangular mesh on top of S. This is much more appropriate in practice, because undersampling, in terms of an r-sampling, occurs inevitably at sharp edges of $\mathcal{O}$. For the other two problems, namely instability and high density of the output, we propose a pruning technique. This technique is based on set-covering, and it extracts a small and stable subset of the initial discrete medial axis transform, which represents $\mathcal{O}$ efficiently. Such a representation of $\mathcal{O}$ is a result by itself, and it can, e.g., be used for collision detection. But it is also particularly well suited for medial axis computations.

Generally, the response of medial axes to even tiny perturbations on an object's surface is unproportional, and we refer to this problem with the term *instability*. Due to instability, one is rather interested in reasonably pruned medial axes than in exact ones. Such approximate medial axes are often computed in two steps: First, a more or less exact medial axis of $\mathcal{O}$ is computed. Then, some pruning criterion is applied in order to remove its superfluous branches. We have reversed these steps: First we

compute a stable union of balls, which efficiently approximates $\mathcal{O}$. Then we compute its exact medial axis as approximation of the medial axis of $\mathcal{O}$. We have implemented the known medial axis algorithm [AK01] to be able to evaluate it in the context of our toolchain, and it works very well. Parts of this work appeared as a conference paper [AAH+07].

1.1.2 Surface reconstruction

Given an unorganized set of points S which are sampled from an unknown surface $\mathcal{F}$ in $\mathbb{R}^3$, construct a surface through the points in S. This is the classical surface reconstruction problem, which has applications in laser scanning and computer graphics. Several surface reconstruction approaches exist, e.g., [ACK01, DG06, BC00] and [CL08], to name a few. Our approach is similar to work in [CL08] in the sense that we also create balls centered at the sample points S, from which we then retrieve the surface. In contrast to [CL08] we do not assume prior knowledge of the local feature sizes of S, but we estimate them using distances from the sample points to the so called *pole points* that occur in the mentioned Voronoi approach. Our work distinguishes also substantially from [CL08] in that we prune the surface balls at a selectable degree before we reconstruct the surface from their union. This way we achieve coarse-to-fine surface reconstruction. We map the omitted sample points to those participating as vertices in the reconstructed mesh and call this enhanced mesh a *seed polytope* of $\mathcal{O}$ because it is intended as a starting point for incremental refinement or reconstruction of $\mathcal{O}$ by, e.g., triangular Bézier patches.

To the best of our knowledge this is the first result that uses, from a practical point of view, approximations of local feature size and medial axis to obtain locally adaptive reconstructions of an unknown surface. We have proven topological correctness of our work, which involved a number of new proofs on angles and distances related to local feature size and pole points. We believe that these proofs are by themselves an important contribution because the mentioned Voronoi machinery has a wide range of applications. Our results appeared as a conference paper [AAK+09].

1.1.3 Minkowski sums

The *Minkowksi sum*, also known as the morphological *dilation* operation, has important applications. Among them are collision detection, motion planning and offset computation. However, the Minkowski sum of two non-convex polyhedra in 3D is inherently

hard to compute. A common decomposition approach works as follows: Decompose the two polyhedra into convex parts. Then compute the pairwise Minkowski sums of these parts, and finally compute the union of the pairwise Minkowski sums. The time complexity of this approach is $O(m^3 \cdot n^3)$, where m and n denote the number of vertices of the two polyhedra, see [FHW08].

We have developed a novel algorithm that computes Minkowski sums of objects, which are represented by unions of balls. In a sense, this input is also a convex representation, and we follow the described scheme in that we compute pairwise Minkowski sums of balls. These are, however, much easier to compute. The union of the pairwise Minkowski sums, whose computation is supported by the power diagram of the balls, serves as approximation of the Minkowski sum of the original objects. We have implemented our approach, and in comparison with other available software packages our application scores well with respect to running time and memory consumption.

1.2 Structure of the thesis

In Chapter 2 we review fundamental data structures and algorithms from computational geometry that we use in our approaches, and we describe a software library that we use for our applications. In Chapter 3 we first review the Voronoi approach [AK00] to compute discrete medial axis transforms from r-sampled objects. Then we describe our hybrid approach of exact and heuristic set-covering methods to prune and stabilize such sets of balls. Finally, we evaluate our implementation of the medial axis algorithm [AK01] in the context of our toolchain. Chapter 4 treats surface reconstruction with adjustable level of detail. Thereby, we enhance the above mentioned Voronoi approach with a number of new theoretical bounds, which enable us to prove topological correctness of our constructions. In Chapter 5, we cover a novel algorithm for approximate Minkowski sums, where the input objects are represented by balls. Finally, Chapter 6 puts the individual algorithms in a more global context and summarizes the contributions of this thesis.

Chapter 2

Background

In this chapter we introduce fundamental data structures and algorithms along with their properties. Among them are triangulations as well as spatial search structures, and we will refer to them throughout this thesis. As a practice related work, we have set value on results that are not only provable but are also feasible in practice. Thus we have implemented most of our approaches. Thereby we relied on the *Computational Geometry Algorithms Library* CGAL [CGA], which provides stable implementations of these algorithms and which we also describe in this chapter.

2.1 Introduction

From a scientific point of view, it makes no sense 'to re-invent the wheel again and again'. Therefore, some of the data structures that we use for our algorithms, have not been implemented from scratch. Instead our software makes use of the robust implementations provided by the *Computational Geometry Algorithms Library* CGAL [CGA]. Although this thesis is *not* primary about CGAL, we start with an overview of CGAL here, because the subsequent descriptions of the fundamental data structures and algorithms will refer to it.

2.1.1 CGAL

The *Computational Geometry Algorithms Library* is a huge library for geometric algorithms and data structures. Currently, CGAL consists of almost 700 000 lines of code, and its user and reference manual comprises more than 4 000 pages. There are also other libraries, which provide adequate implementations for our needs. But CGAL is open source software, it has a uniform interface and it is supported by an active community. These advantages make it the best choice for scientific implementations.

The CGAL library is released under different licenses. Our software uses the so called *linear kernel* which is under the *GNU Lesser General Public License [Fre01]*, as well as the packages mentioned in the following, which are under the *Q public license [Tro]*.

2.1.2 The linear kernel in CGAL

An issue of many implementations of geometric algorithms is computational accuracy. Floating point arithmetic with limited precision data types like, e.g., in C++ the built-in data type *double*, is afflicted with numerical errors. While tiny geometric inaccuracies introduced by these errors might be acceptable, a program might hang or crash when it relies on predicates, which are evaluated using inexact arithmetic, because a wrong result can lead to a wrong decision or it can lead a program into a wrong state.

To achieve robust implementations, we use a so called *linear kernel* from CGAL. In CGAL, a kernel consists of a collection of basic geometric objects like points, lines, triangles etc., construction operations like the intersection of two objects, and predicates like, e.g., orientation tests. CGAL follows the generic programming paradigm, and thus the used number type can be chosen according to the needs of a specific problem. We use two different, predefined CGAL kernels for our implementations, and their behavior

is determined by nested chains of parametrizations, see Listing 2.1.

Listing 2.1: Parametrization of CGAL Kernels

```
typedef Filtered_kernel< Simple_cartesian<double> >
        Exact_predicates_inexact_constructions_kernel;
typedef Lazy_kernel<Simple_cartesian<Gmpq> >
        Exact_predicates_exact_constructions_kernel;
```

As to the *Exact_predicates_inexact_constructions_kernel*: The part that consists of the template class *Simple_cartesian*, parametrized by the C++ number type *double*, denotes already a simple CGAL kernel. But this kernel uses cheap (and inexact) double precision floating point arithmetic for both, evaluation of predicates and representation of objects by cartesian coordinates. To achieve exact evaluation of predicates, one could parametrize this kernel with a slow multi-precision number type instead of *double*. But there is a more efficient way to achieve robustness, which is called arithmetic filtering: The inexact kernel *Simple_cartesian<double>* is plugged into a *Filtered_kernel* [FS06], which uses interval arithmetic [BBP01] to detect if a predicate, evaluated with the inexact number type, is possibly incorrect. In cases where the arithmetic filter fails, the predicate is re-evaluated with arbitrary-precision rational arithmetic. Thus, the filtered kernel provides exact predicates although it uses computationally expensive multi-precision arithmetic (hopefully) only rarely. The major drawback of the *Exact_predicates_inexact_constructions_kernel* is that its constructions are computed with inexact arithmetic, such that even its exactly evaluated predicates will possibly fail if they involve constructed objects, like shown in Listing 2.2.

Listing 2.2: Provoking an error in spite of exact predicates

```
typedef CGAL::Exact_predicates_inexact_constructions_kernel K;
typedef K::Point_2 Point;
typedef K::Line_2 Line;

Line line1(Point(0,0),Point(1,2));
Line line2(Point(0,1),Point(1,0));
Point p;
assign(p,intersection(line1,line2));
assert(line1.has_on(p) && line2.has_on(p));
```

Therefore, in cases where exact constructions are necessary to make our implementations robust, we use the *Exact_predicates_exact_constructions_kernel*, whose parametrization is also shown in Listing 2.1. For this kernel, the template class *Simple_cartesian*

is parametrized differently, namely by the class *Gmpq*, which provides an arbitrary precision rational number type based on the GNU Multiple Precision Arithmetic Library [GMP]. Although *Simple_cartesian<Gmpq>* denotes already an exact kernel, it is, for performance considerations, not used directly. Instead this kernel is plugged into a *Lazy_kernel* which tentatively uses interval arithmetic for constructions. If later a filter fails, i.e. a predicate, applied to an object, can not be evaluated safely, then exact arithmetic is used. The additional cost of this kernel is an increased memory usage as the history of all construction steps for the inexactly evaluated objects has to be stored.

We refrain from repeating a huge number of details from the manual and refer the interested reader to [CGA].

2.2 Voronoi diagram

The Voronoi diagram [Aur91] of a set S of points (sites) in $\mathbb{R}^3$ is a partition of space into (possibly unbounded) polyhedral regions, called Voronoi cells. Several types of Voronoi diagrams - depending on the underlying metric - exist, and the most common one is the euclidean nearest site Voronoi diagram to which we refer in this thesis whenever we use the term Voronoi diagram. Let $d(a, b)$ denote the euclidean distance between the two points a and b in $\mathbb{R}^3$. In the Voronoi diagram $\mathrm{V}(S)$ of S, each site $s_i \in S$ has an associated Voronoi cell $\mathrm{v}(s_i)$ which consists of all points of $\mathbb{R}^3$ having s_i as their nearest site:

$$\mathrm{v}(s_i) = \{p \mid d(p, s_i) \leq d(p, s_j),\ s_i, s_j \in S\}$$

These Voronoi cells are convex, and there is exactly one cell for each site $s \in S$, which is infinite iff s is an element of the convex hull of S. A remarkable property of the Voronoi diagram is its duality to the Delaunay triangulation: In CGAL, 3D Voronoi diagrams can not be computed explicitly. Instead, the Delaunay triangulation DT of the input points must be computed, and the elements of the Voronoi diagram can then be retrieved as the dual elements of DT as described in Section 2.3.

Concerning the complexity of the Voronoi diagram, let n be the number of points in S. Then each of the n regions of $\mathrm{V}(S)$ can share a Voronoi facet with the other $n-1$ regions. In $\mathbb{R}^3$ worst case configurations of S exist (see [DV77]) that enforce the maximum number of $\binom{n}{2}$ facets in $\mathrm{V}(S)$, so the complexity in $\mathbb{R}^3$ is $O(n^2)$. However, Dwyer [Dwy89] showed that the expected size of $\mathrm{V}(S)$ in $\mathbb{R}^d$ is $O(n)$ if S is a random point set uniformly distributed in the unit ball. In our setting, the sites are points from the boundary of three-dimensional objects, and from an empirical point of view, both

the size of their Voronoi diagrams as well as the computational runtime, behave also linear.

2.3 Delaunay triangulation

The Delaunay triangulation DT(S) [Aur91] of a point set S is the dual diagram of the euclidean nearest site Voronoi diagram of S. In $\mathbb{R}^3$, each m-face of one diagram corresponds to an n-face of the other diagram, such that $m + n = 3$ holds:

- Delaunay tetrahedron $\Leftrightarrow$ Voronoi vertex
- Delaunay facet $\Leftrightarrow$ Voronoi segment
- Delaunay segment $\Leftrightarrow$ Voronoi facet
- Delaunay vertex $\Leftrightarrow$ Voronoi cell

There is an implementation for Delaunay triangulations in CGAL, and CGAL provides so called circulators (which are basically iterators that work in a circular fashion) that can be used to traverse the elements of DT(S) conveniently in order to compute their dual elements in V(S). For example, the Voronoi facet f_V which is dual to a Delaunay edge e_D can be computed by retrieving all Delaunay tetrahedra incident to e_D and computing their circumcenters which are just the vertices of the facet f_V.

Along with the duality to the Voronoi diagram, one of the most remarkable properties of the Delaunay triangulation is its empty-sphere property. The circumsphere of each tetrahedron in DT(S) is empty, i.e., no point of S is contained in the circumsphere of any tetrahedron of the Delaunay triangulation. We will use this property later on in many of our proofs and constructions.

2.4 Power diagram

A power diagram [Aur88] is a generalized Voronoi diagram, computed on top of a set of weighted points $S_w = \{(p_i, w_i) | i = 1, \cdots, n\}$, where p_i denotes a point site and w_i is its weight. Let

$$\pi((p_i, w_i), (p_j, w_j)) = \|p_i - p_j\|^2 - w_i - w_j \tag{2.1}$$

denote the power distance of two weighted points $s_i = (p_i, w_i)$ and $s_j = (p_j, w_j)$. The power diagram of S_w is a partition of space into a collection of possibly unbounded

polyhedral cells, defined by

$$cell(s_i) = \{x \in \mathbb{R}^{\mathrm{d}} \,|\pi(x, s_i) \leq \pi(x, s_j),\ s_i \text{ and } s_j \in S_w\}$$

The Voronoi diagram can be seen as a special case of power diagrams where all weights are 0. But unlike in Voronoi diagrams, cells of a power diagram do not need to contain their weighted site, and moreover, sites do not need to have non-empty cells at all. Power diagrams have remarkable properties with respect to unions of balls, because balls can be interpreted as weighted points, see Chapter 5.

As in the case of the Voronoi diagram, CGAL does not support explicit computation of the power diagram. Instead, the power diagram must be derived from its dual diagram, which is a regular triangulation, see Section 2.5.

2.5 Regular triangulation

Like the Delaunay triangulation of an unweighted point set S is dual to the Voronoi diagram of S, a regular triangulation of a weighted point set S_w is dual to the power diagram of S_w. In $\mathbb{R}^3$, a subset $T \in S_w$ with $|T| = 4$ forms a tetrahedron of the regular triangulation of S_w, iff there is an unweighted point $x \in \mathbb{R}^3$, such that the power distance of x with respect to the points in T is equal and minimal with respect to the points in $S_w \setminus T$. Unlike in Delaunay triangulations, possibly only a subset of S_w is involved in its regular triangulation. The Delaunay triangulation is a special case of regular triangulations, where all weights are 0.

In CGAL, the power diagram of S_w can be computed from the regular triangulation of S_w using the duality already described in Section 2.3.

2.6 Weighted α-shape

The weighted α-shape, [Ede92], $\mathcal{A}_\alpha(S_w)$ of a set of weighted points S_w in $\mathbb{R}^3$ is a polytope in $\mathbb{R}^3$, that is uniquely defined by S_w and a value $\alpha \in \mathbb{R}$. Consider a subset $T \subseteq S_w$ with $|T| = k + 1 \leq 3$. T spans a k-simplex σ_T, which is called α-exposed if there exists a weighted point (q, α) such that (see Equation 2.1)

$$\pi((p, w), (q, \alpha)) \begin{cases} = 0 & \forall (p, w) \in T \\ > 0 & \forall (p, w) \in S_w \setminus T \end{cases}$$

The boundary of $\mathcal{A}_\alpha(S_w)$ is the union of all α-exposed simplices spanned by subsets of S_w (Definition adapted from [Ede92]). Only the weighted zero α-shape, i.e., the

weighted α-shape for $\alpha = 0$, is used in this thesis, and we will, for short, denote $\mathcal{A}_0(S_w)$ by $\mathcal{A}(S_w)$.

2.7 kd-tree

A kd-tree [Ben75] is a binary tree that organizes a set of k-dimensional points such that geometric queries can be performed efficiently. It subdivides $\mathbb{R}^k$ into axis aligned cells. The root node of the tree consists of a cell which encloses all points. It is recursively split by a hyperplane orthogonal to its longest coordinate axis at the median of the data, and the two new cells are the sons of the node that has been split. The subdivision is stopped as soon as the number of data points in a cell is below a certain threshold. There are several variants of such trees, and they vary with respect to the splitting rule (where to split), the choice of the coordinate axis (one can cycle through the coordinate axes or always choose the longest one) and the abort criterion, i.e. the threshold or bucket-size. kd-trees can be built in $O(n \cdot log(n))$ time, and they require $O(n)$ memory. We use CGAL's kd-tree in Chapter 3 when we perform spherical range queries, i.e. when we search for points being contained in a certain sphere.

2.8 Skin surfaces

The algorithms described in this thesis treat objects by unions of balls. But sometimes it might be desirable to convert intermediate results or the final output of our software to triangular meshes. Fortunately, the so called *Skin Surface* package [Kru09] is contained in CGAL, which allows to compute from a given set of balls a triangular mesh which approximately represents the boundary of its union. We believe that the possibility to do such ball-to-mesh conversions greatly enhances the value of our techniques.

Chapter 3

Medial Axis Transforms and Medial Axes

3.1 Motivation

It is simple and intuitive to describe a three-dimensional object $\mathcal{O}$ by its boundary, e.g., by a polygonal mesh. This representation is beneficial for visualization, but further processing with $\mathcal{O}$ often requires adapted representations. Collision detection, for example, is faster with octree or sphere-tree representations [BO04]. Similarly, a number of other algorithms takes advantage of a representation which approximates $\mathcal{O}$ by the union of a set of balls, the so-called *discrete medial axis transform* $\mathrm{DMT}(\mathcal{O})$. Aside from collision detection we have fast Minkowski sums, medial axes and surface reconstruction in mind. Nevertheless, algorithms using this kind of representation are hardly seen in practice because it is hard to compute a stable set $\mathrm{DMT}(\mathcal{O})$ with low cardinality.

Medial axes are important shapes in computational geometry and computer graphics as they allow for object simplification, mesh generation and analysis of an object. But instability is inherent to medial axes in the sense that they respond unproportionally to insignificant features of the object surface. Thus, one is rather interested in stable approximations [DZ04, FLM03].

We propose a practically approved approach to compute a stable and small discrete medial axis transform from a boundary triangulated object $\mathcal{O}$. This result enables us to compute stable, piecewise linear approximations of the medial axis of $\mathcal{O}$ with the algorithm [AK01] as demonstrated in this chapter.

3.2 Introduction

The continuous *medial axis transform* $\mathrm{MT}(\mathcal{O})$ of an object $\mathcal{O}$ is an infinite set of balls, whose union is just $\mathcal{O}$, see Definition 3.1. These balls are medial balls, i.e., they are centered at the medial axis $\mathrm{M}(\mathcal{O})$. Unfortunately, a large fraction of $\mathrm{M}(\mathcal{O})$ typically consists of branches which correspond to insignificant features or even noise on the surface of $\mathcal{O}$. In $\mathrm{MT}(\mathcal{O})$, this unstable behavior is expressed by many small, surface-near balls which describe the same insignificant features.

The first goal described in this chapter is to replace $\mathrm{MT}(\mathcal{O})$ by a finite set of balls $\mathrm{DMT}(\mathcal{O})$, whose union approximates $\mathcal{O}$. We started to tackle the problem with the theoretically well founded work [AB99, ACK01] to which we refer as *Voronoi approach*. The Voronoi approach yields such a set $\mathrm{DMT}(\mathcal{O})$ from a dense point sample S of $\mathcal{O}$, see Definition 3.2. But in practice we faced several difficulties with the original Voronoi approach. First of all, the Voronoi approach requires an r-sample of a smooth object as input, and it does hardly work properly otherwise. Our extension to overcome this issue is of a practical nature. Our implementation takes a triangular mesh τ as input, see [AAH$^+$07]. The mesh allows our variant to work correctly even if S (which consists of the vertices of τ in our case) is not r-sampled. The second problem of the Voronoi approach is over-density. As observed in practice, the cardinality of $\mathrm{DMT}(\mathcal{O})$ is always in the range of $|S|$, and thus $\mathrm{DMT}(\mathcal{O})$ is much more dense than required in our setting. Finally, the key problem of the original Voronoi approach is the high accuracy by which it models the unstable behavior of $\mathrm{MT}(\mathcal{O})$. Thus, $\mathrm{DMT}(\mathcal{O})$ contains many small, surface-near balls to which we refer as *unstable balls*. Several pruning criteria have been proposed to eliminate the unstable balls from $\mathrm{DMT}(\mathcal{O})$, [SFM07, SAAY06, FLM03, SB98]. But still the question remains open how to lower the density of the stable balls of $\mathrm{DMT}(\mathcal{O})$.

We propose a natural pruning scheme, based on set covering, to tackle two problems at once, instability and over-density of $\mathrm{DMT}(\mathcal{O})$. Thereby we generate from $\mathrm{DMT}(\mathcal{O})$ a new set $\mathrm{DMT}^+_{\mathrm{in}}$ of slightly enlarged balls which covers S redundantly. Then we choose from $\mathrm{DMT}^+_{\mathrm{in}}$ a preferably minimal subset $\mathrm{DMT}^*_{\mathrm{in}}$ which still covers S. This is an instance of the classical *set covering problem*. The set covering aims to minimize $|\mathrm{DMT}^*_{\mathrm{in}}|$. To this end it must prefer balls which are centered near stable parts of $\mathrm{M}(\mathcal{O})$ because they cover larger subsets of S than unstable balls. Thus the goal of stabilization is implicitly reached. In this approach, the degree of pruning can be chosen via the ball enlargement. Low density and stability make $\mathrm{DMT}^*_{\mathrm{in}}$ an ideal input for the elegant

medial axis algorithm [AK01] which operates on unions of balls. We have implemented this algorithm to demonstrate the potential of our novel method to stabilize medial axes.

The rest of this chapter is organized as follows: In Section 3.3 we recall fundamental definitions and introduce the used notation. Section 3.4 deals with the original Voronoi approach. In Section 3.5 we describe two different methods to compute $\mathrm{DMT}^{+}_{\mathrm{in}}$ in a topologically correct way. In Section 3.6 we describe our hybrid technique of exact and heuristic set covering methods. Moreover a post-processing step that ensures topological correctness of $\mathrm{DMT}^{*}_{\mathrm{in}}$ is described. In Section 3.7 we shortly sketch the algorithm [AK01]. Section 3.8 covers important details of our implementation. Finally, in Section 3.9 we use three different models to demonstrate how our constructions work in practice.

3.2.1 Related work

Aside from the already mentioned related work from Amenta et al. [AB99, ACK01, AK01], we are aware of [BO04]. In this work, sphere-trees for hierarchical collision detection are constructed. The authors use set covering implicitly and they do not create balls via poles. In contrast we develop the set covering approach systematically and for a different goal, namely to stabilize medial axes. Thereby we we achieve more efficient and topologically correct representations. Therefore we claim that set covering for medial axis stabilization is a new contribution.

3.3 Definitions and notation

Let $\mathcal{O}$ denote a bounded set in $\mathbb{R}^3$, and denote with $\mathcal{F}$ its boundary. We will assume that $\mathcal{F}$ is a connected set. The definitions below are standard in the literature; see, e.g., [BT06].

Definition 3.1.

- *The* medial axis transform, $\mathrm{MT}(\mathcal{O})$, *of* $\mathcal{O}$ *is the (infinite) collection of maximal balls that avoid* $\mathcal{F}$*, where maximality is with respect to ball inclusion.*
- *The* medial axis, $\mathrm{M}(\mathcal{O})$, *of* $\mathcal{O}$ *is the set of centers of the balls in* $\mathrm{MT}(\mathcal{O})$.

The surface $\mathcal{F}$ splits $\mathrm{M}(\mathcal{O})$ and $\mathrm{MT}(\mathcal{O})$ in their inner and outer parts. $\mathrm{M}(\mathcal{O})$ behaves unstable in the sense that it responds unproportionally to even tiny features of $\mathcal{F}$. The

corresponding medial axis transform $\mathrm{MT}(\mathcal{O})$ expresses this instability by describing those tiny features by surface-near balls.

Definition 3.2.

- *The* local feature size $\mathrm{lfs}(x)$ *of a point* $x \in \mathcal{F}$ *is the minimum distance from* x *to any point of* $\mathrm{M}(\mathcal{O})$.
- *A finite point set* $S \subset \mathcal{F}$ *is an* r-sample *of* $\mathcal{F}$ *if every point* $x \in \mathcal{F}$ *has at least one point* $s \in S$ *within distance* $r \cdot \mathrm{lfs}(x)$.
- *The* Voronoi diagram *of a finite set* S *of points in* $\mathbb{R}^3$ *is a partition of* $\mathbb{R}^3$ *where each point* $s_i \in S$ *has its associated Voronoi cell*

$$V(s_i) = \{x \in \mathbb{R}^3 : \|x - s_i\| \leq \|x - s_j\|, s_i, s_j \in S\}$$

- *For a (sample) point set* $S \subset \mathcal{F}$*, the two vertices of a Voronoi cell* $V(s_i)$ *being farthest away from* $s_i \in S$ *on either side are called the* poles *of* $V(s_i)$.

The notions of r-samples and poles have been introduced by Amenta and Bern in [AB99]. Throughout this paper, we assume that S is an r-sample of $\mathcal{F}$ for $r = 0.08$.

3.4 Discrete medial axis (transform)

For practical purposes, it is beneficial to compute a finite subset of $\mathrm{MT}(\mathcal{O})$. We will refer to such a subset as a *discrete medial axis transform*, $\mathrm{DMT}(\mathcal{O})$, of $\mathcal{O}$. The medial axis of the union of all the balls selected for $\mathrm{DMT}(\mathcal{O})$ is called the *discrete medial axis*, $\mathrm{DM}(\mathcal{O})$, of $\mathcal{O}$. It is well known that $\mathrm{DM}(\mathcal{O})$ is a piecewise linear object that can be constructed efficiently once the set $\mathrm{DMT}(\mathcal{O})$ is given [AM97, AK01, GMP07].

In [AK00], Amenta and Kolluri describe a Voronoi diagram based approach to compute such a discrete medial axis transform which works as follows. Let S denote some set of sample points from $\mathcal{F}$. First, the Voronoi diagram of S is computed from which all pole points are extracted. Then the discrete medial axis transform $\mathrm{DMT}(\mathcal{O})$, consisting of so called *polar balls*, is constructed. A polar ball is a ball centered at some pole point p, and its radius is the euclidean distance between p and the related site $s \in S$. The surface $\mathcal{F}$ of the object represented by S splits the set of polar balls into two parts, where the *inner part* approximates the object and the *outer part* approximates its complement. As shown in [AK00], the inner part, $\mathrm{DMT}_{\mathrm{in}}$, is homeomorphic to the

original object $\mathcal{O}$, provided S is an r-sample of $\mathcal{O}$ (for suitable r). This implies that the corresponding discrete medial axis, $\mathrm{DM}_{\mathrm{in}}$, is homotopy equivalent to $\mathcal{O}$. Pole points are labeled as inner or outer poles according to an angle criterion which works for r-sampled objects. As an alternative [AAH+07], an additional triangular mesh on top of S can be used, in order to overcome labeling problems if S fails to be an r-sampling, for example, when $\mathcal{F}$ contains sharp edges or is poorly sampled and noisy.

By construction, each Voronoi vertex is either not a pole point, or it is a pole point for k sample points with $1 \leq k \leq 4$ in the non-degenerate case, which implies

$$\frac{|S|}{4} \leq |\mathrm{DMT}_{\mathrm{in}}| \leq |S| \ .$$

From computations with real data sets we know that $|\mathrm{DMT}_{\mathrm{in}}|$ is typically almost $|S|$. Thus, $\mathrm{DMT}_{\mathrm{in}}$ is very dense and, moreover, as it approximates the inner part of the medial axis transform $\mathrm{MT}_{\mathrm{in}}(\mathcal{O})$, it features the same instability.

3.5 Ball enlargement

We additively enlarge the radii of the balls b_i in $\mathrm{DMT}_{\mathrm{in}}$ by small values $\varepsilon_i > 0$ and refer to the set of enlarged balls as $\mathrm{DMT}^+_{\mathrm{in}}$. The size of ε_i controls the degree by which $\mathrm{DMT}^+_{\mathrm{in}}$ will be pruned in the subsequent set covering step. Note that multiplicative enlargement is an option as well, but the choice for additive enlargement reflects our intention that noise and small features should have the same (in-)significance everywhere on $\mathcal{F}$. Whatever method is chosen, ε_i can not be made arbitrarily large, because we demand the unions of the balls in $\mathrm{DMT}_{\mathrm{in}}$ and $\mathrm{DMT}^+_{\mathrm{in}}$, respectively, to have the same topology. Denote with $\mathrm{DM}_{\mathrm{in}}$ (and $\mathrm{DM}_{\mathrm{out}}$) the medial axis of the union of all the balls in $\mathrm{DMT}_{\mathrm{in}}$ (and $\mathrm{DMT}_{\mathrm{out}}$).

3.5.1 Pole method

As a necessary topology condition, balls from different sides of $\mathrm{DM}_{\mathrm{out}}$ must not intersect. To this end, we compute for each ball $b \in \mathrm{DMT}_{\mathrm{in}}$ a bound on its radius enlargement such that b still avoids $\mathrm{DM}_{\mathrm{out}}$. We start with a technical lemma. Recall that an r-sampling S of the object boundary $\mathcal{F}$ is available.

Lemma 3.3. *Let $b_{p,\rho'} \in \mathrm{DMT}^+_{\mathrm{in}}$ be a ball with radius ρ', centered at an inner pole point p. For every point $y \in \mathrm{DM}_{\mathrm{in}} \cap b_{p,\rho'}$ the line segment $\overline{py}$ intersects $\mathcal{F}$ in at least*

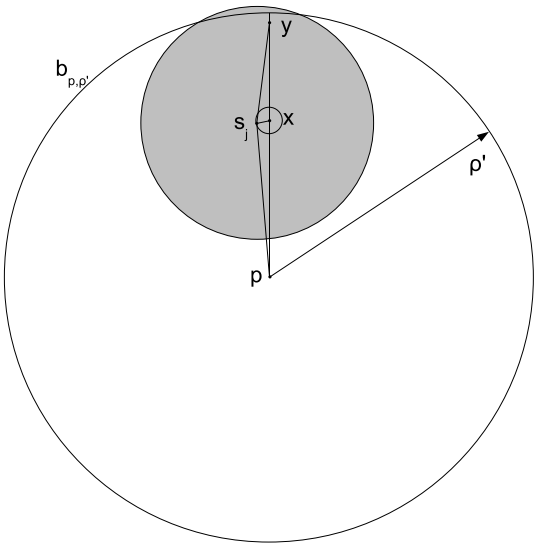


Figure 3.1: Topological correctness

one point x, and the distance between p and the closest sample point $s_j \in S$ of x is

$$\|s_j - p\| \leq \frac{1-r}{1-2.2802 \cdot r} \cdot \rho' \underset{r \leq 0.08}{<} 1.126 \cdot \rho'$$

Proof. Because S is an r-sampling of $\mathcal{F}$, we have

$$\|s_j - x\| \leq r \cdot \mathrm{lfs}(x) \ .$$

Due to the Lipschitz-continuity of local feature sizes [AB99, Lemma 1], we know

$$\mathrm{lfs}(x) \leq \frac{\mathrm{lfs}(s_j)}{1-r} \ .$$

This combines to

$$\|s_j - x\| \leq \frac{r}{1-r} \cdot \mathrm{lfs}(s_j) \ .$$

Let $\hat{D}(s_j)$ denote the distance from a sample point s_j to its closest pole point. In [AAK$^+$09], Lemma 5.1, we prove an upper bound for the local feature size of s_j, namely $\mathrm{lfs}(s_j) \leq 1.2802 \cdot \hat{D}(s_j)$. Thus

$$\|s_j - x\| \leq 1.2802 \cdot \frac{r}{1-r} \cdot \hat{D}(s_j) \ . \tag{3.1}$$

As y is contained in $b_{p,\rho'}$ and x lies on the segment $\overline{py}$ we have

$$\|p - x\| \leq \rho' \ .$$

By the triangle inequation (see Figure 3.1) we now get

$$\begin{aligned}\|s_j - p\| &\leq \|p - x\| + \|s_j - x\| \\ &\leq \rho' + 1.2802 \cdot \frac{r}{1-r} \cdot \hat{D}(s_j) \\ &\leq \rho' + 1.2802 \cdot \frac{r}{1-r} \cdot \|s_j - p\| \ , \\ \|s_j - p\| &\leq \frac{\rho'}{1 - 1.2802 \cdot \frac{r}{1-r}} = \frac{1-r}{1 - 2.2802 \cdot r} \cdot \rho' \ .\end{aligned}$$

□

The next lemma uses the bound developed in Lemma 3.3 to define a critical range around each polar ball in which we consider the contained sample points.

Lemma 3.4. *Let $b_{p,\rho'} \in \mathrm{DMT}^+_{\mathrm{in}}$ be a ball with radius ρ', centered at an inner pole point p. Let $S' = \{s \in S : \|s - p\| \leq 1.126 \cdot \rho'\}$ and let $\hat{D}(s_i)$ denote the distance from $s_i \in S'$ to the closest pole point of s_i. Then $b_{p,\rho'} \cap \mathrm{DM}_{\mathrm{out}} = \emptyset$ if*

$$\rho' < \min_{s_i \in S'} (\|p - s_i\| + (0.817 - 1.2802 \cdot \frac{r}{1-r}) \cdot \hat{D}(s_i)) \tag{3.2}$$

Proof. To obtain a contradiction, we assume that $b_{p,\rho'} \cap \mathrm{DM}_{\mathrm{out}} \neq \emptyset$ and let y be the closest point of $b_{p,\rho'} \cap \mathrm{DM}_{\mathrm{out}}$ to p. We have the same setting as in the proof of Lemma 3.3: There is a point $x \in \mathcal{F}$ where $\overline{py}$ intersects $\mathcal{F}$, and as shown in Equation 3.1, the distance to its nearest pole point s_j is bounded by

$$\|s_j - x\| \leq 1.2802 \cdot \frac{r}{1-r} \cdot \hat{D}(s_j) \ . \tag{3.3}$$

By assumption, $\|p - y\| \leq \rho'$, and the points p, x and y are collinear, so we get

$$\|x - y\| \leq \rho' - \|p - x\| \ .$$

By the triangle inequation (see Figure 3.1) we have

$$\begin{aligned}\|x - y\| &\leq \rho' - \|p - s_j\| - \|s_j - x\| \\ &\leq \rho' - \|p - s_j\| \ .\end{aligned}$$

Substituting ρ' from Equation 3.2 we get

$$\|x - y\| \leq \min_{s_i \in S'} (\|p - s_i\| + (0.817 - 1.2802 \cdot \frac{r}{1-r}) \cdot \hat{D}(s_i)) - \|p - s_j\| \ .$$

As $\hat{D}(s_i) \leq \|p - s_i\|$, the point s_i which minimizes $\min_{s_i \in S'}(\|p - s_i\| + (0.817 - 1.2802 \cdot \frac{r}{1-r}) \cdot \hat{D}(s_i))$ is the one which realizes $\min_{s_i \in S'} \|p - s_i\|$. Thus $\|p - s_i\| \leq \|p - s_j\|$ and we can write

$$\|x - y\| < (0.817 - 1.2802 \cdot \frac{r}{1-r}) \cdot \hat{D}(s_j) \ .$$

Together with Equation 3.3 we have

$$\|x - y\| < 0.817 \cdot \hat{D}(s_j) - \|s_j - x\| \ . \tag{3.4}$$

In [AAK+09], we introduce the discrete local feature size $\widetilde{\text{lfs}}(s)$ of a sample point s as its distance to the closest point of the discrete medial axis $\text{DM}(\mathcal{O})$ and prove that $\widetilde{\text{lfs}}(s) \geq 0.817 \cdot \hat{D}(s)$. By the triangle inequation we have

$$\|s_j - y\| \leq \|s_j - x\| + \|x - y\|$$

We substitute Equation 3.4 and get

$$\|s_j - y\| < \|s_j - x\| + 0.817 \cdot \hat{D}(s_j) - \|s_j - x\|$$

$$\|s_j - y\| < 0.817 \cdot \hat{D}(s_j) \ .$$

This is a contradiction, because $\widetilde{\text{lfs}}(s_j) \geq 0.817 \cdot \hat{D}(s)$, and thus the lemma is proven. □

Corollary 3.5. *By Lemma 3.4, we can determine for each polar ball $b_{p,\rho}$ a new radius $\rho' > \rho$, such that the enlarged ball $b_{p,\rho'}$ does not intersect* DM_{out}.

Ensuring that DM_{out} and DMT^+_{in} remain disjoint will give correct results in practice. However, we can construct examples where DMT^+_{in} defines holes and tunnels that are not present in DMT_{in} even if this condition is maintained. To overcome this problem, we can, as an alternative, utilize the power diagram, $\mathcal{PD}(\text{DMT}_{\text{in}})$, of the balls in DMT_{in}.

3.5.2 Power diagram method

$\mathcal{PD}(\text{DMT}_{\text{in}})$ is the power diagram [Aur88] of the set of weighted points that we get by considering the centers of the balls in DMT_{in} and weighting them by their squared radii. This diagram contains a (possibly empty) polyhedral cell for each weighted point. For the corresponding ball, exactly those parts which contribute to the union of all the balls in DMT_{in} are contained in this cell. The diagram does not change when the weights of all its points are additively enlarged by the same value. If we enlarge the respective balls in this manner, topological changes for DMT_{in} will occur only if a group of balls

(of size two, three, or four, in the non-degenerate case) gets mutually in touch. But using only this property would lead to lots of false positives, as such new intersections occur also frequently within the union of the balls to be enlarged, which then do not harm.

Definition 3.6. *Let f be a face (facet, edge, or vertex) of $\mathcal{PD}(\mathrm{DMT}_{\mathrm{in}})$, and consider the intersection, $g(f)$, of f with its dual face (edge, triangle, or tetrahedron) in the corresponding regular triangulation. If $g(f) \neq \emptyset$ then we call this point the* Gabriel point *of f. We say that $g(f)$ is* critical *if $g(f)$ is not contained in any ball from $\mathrm{DMT}_{\mathrm{in}}$.*

If we enlarge the balls in $\mathrm{DMT}_{\mathrm{in}}$ as described above, then the critical Gabriel points are just the points where a face is touched first (and simultaneously) by its defining balls. We use this property for a growing algorithm which constructs $\mathrm{DMT}_{\mathrm{in}}^{+}$ as follows.

Step 1 Compute $\mathcal{PD}(\mathrm{DMT}_{\mathrm{in}})$ and its set G of critical Gabriel points.

Step 2 Enlarge all weights additively by a common value, such that G and the enlarged set of balls remain disjoint. If the enlargement meets the user's choice, we output the current set as $\mathrm{DMT}_{\mathrm{in}}^{+}$. Otherwise, we continue with the next step.

Step 3 Let $g \in G$ denote the point which has stopped the enlargement in Step 2. We keep the (at most four) balls associated to g at constant size for the rest of the algorithm, and remove their cells from the power diagram. (The set G of Gabriel points is kept unchanged.)

Continue with Step 2.

Concerning the correctness of the algorithm, note that ceasing the growth of balls cannot cause undetected intersections. Moreover, it is not necessary to increase G with additional critical Gabriel points that would stem from updating the power diagram after deletion of ceased balls: Such points have to lie in the interior of initial power cells, and thus are never touched prior to points already in G.

Observe that Gabriel points on power facets prevent new tunnels from being created, whereas Gabriel points on power edges prevent existing tunnels from disappearing, and new holes from appearing. Finally, Gabriel points at vertices of the power diagram ensure that no already existing hole is covered.

```
Procedure GreedySetcovering
DMT*_in = ∅
while(S ≠ ∅)
{
        Let ~b_i ∈ DMT+_in be a ball which maximizes ~|b_i ∩ S|
        DMT*_in = DMT*_in ∪ b_i
        S = S \ b_i
}
```

Listing 3.1: Simple greedy set covering algorithm

3.6 Pruning the set of balls

Our next goal is to extract a (preferably minimal) subset $\mathrm{DMT}^*_{\mathrm{in}} \subset \mathrm{DMT}^+_{\mathrm{in}}$ such that the balls in this subset still cover all sample points in S. This is an instance of the well known set covering problem. The general set covering problem is NP-hard [Kar72]. The input from our specific setting might be easier, but exact computations for more than 400 balls are unlikely to finish within reasonable time, while we need solutions for a thousand times that many.

There is a standard greedy heuristic which computes approximate solutions, see Listing 3.1. This algorithm comes with a theoretically best-possible approximation guarantee, stating that the output is at most $H(k)$ times larger than the optimal solution, where k is the number of covering sets (balls in our case) and $H(k) \leq \ln k + 1$ is the k-th harmonic number.

We have implemented an elaborate combination of exact and heuristic methods which computes not only an approximate solution for the set covering problem, but also an upper bound on its overhead compared to the optimal solution. We obtain significantly better results than with the simple greedy algorithm; see Section 3.9.

3.6.1 Set covering

For the subsequent set covering step, we first transform geometric properties into combinatorial information. We compute a so-called covering matrix $\mathbf{C}$, which is an $(k \times n)$-matrix with $k = |\mathrm{DMT}^+_{\mathrm{in}}|$ and $n = |S|$. For a ball $b_i \in \mathrm{DMT}^+_{\mathrm{in}}$ and a sample point $s_j \in S$, the entry in $\mathbf{C}$ is defined as

$$c_{i,j} = \begin{cases} 1 & \text{if } s_j \in b_i \\ 0 & \text{otherwise.} \end{cases}$$

$\mathbf{C}$ can be computed within reasonable time using an efficient spatial search structure based on kd-trees. Typically $\mathbf{C}$ is quite sparse, but the number of 1-entries in the matrix depends not only on the amount of enlargement of the balls, but also on geometric properties of the object $\mathcal{O}$ to be approximated. As an extreme case, when $\mathcal{O}$ is a ball itself, we will get a matrix of ones, even for tiny enlargements. Fortunately, excessive memory consumption can nevertheless be avoided using certain reduction rules.

The covering matrix $\mathbf{C}$ is likely to contain redundancy, in the sense that after its removal the reduced problem still grants an optimal solution. There are three reduction rules that are standard in the field of FPGA design, see [CMF93], where logic minimization of boolean functions involves set covering. We say that a row (column) dominates another one if it includes the same 1-entries.

- If row i is dominated by row k, then row i (ball b_i) can be deleted from $\mathbf{C}$ because all points covered by b_i are also covered by ball b_k.
- If column j dominates column ℓ, then column j can be deleted from $\mathbf{C}$ because every solution that covers point p_ℓ will cover point p_j as well.
- If column j has a single 1-entry, e.g., in row i, then this row can be deleted from $\mathbf{C}$. Ball b_i is necessarily contained in any solution, and is marked as part of the result.

These reduction rules are applied iteratively as long as they are successful. In our experiments, the number of 1-entries in $\mathbf{C}$ is significantly reduced after this step.

3.6.2 Exact partial solutions

We continue with integer programming (IP) in order to keep optimality as long as possible. Let $x = (x_1, \ldots, x_k)$ where $k = |\mathrm{DMT}^+_{\mathrm{in}}|$. The IP formulation for the set covering problem is

$$\min \sum x_j \text{ w.r.t. } \mathbf{C}^T \cdot x \geq (1, \ldots, 1)^T, x_j \in \{0, 1\}. \tag{3.5}$$

Although integer programming is NP-hard, moderately sized problems can be solved with a branch-and-bound technique; see [LPS]. After reduction according to the rules above, we split our data into independent parts (submatrices) $\mathbf{C}_i$ of $\mathbf{C}$, where independency is given by the connected components of the corresponding bipartite incidence graph between points and balls. Parts $\mathbf{C}_i$ being small enough are then solved exactly and removed from $\mathbf{C}$, and the balls associated with the (optimal partial) solution vector x are marked as such.

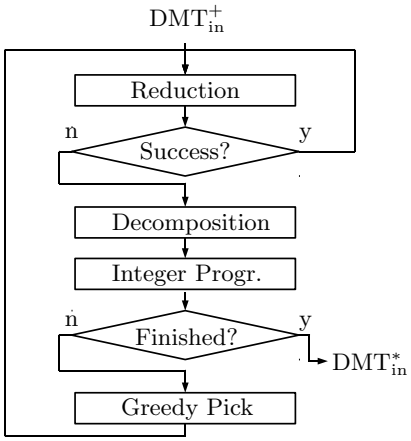


Figure 3.2: Control flow of the set covering algorithm

3.6.3 Greedy selection

Whereas up to this point optimality is kept, we have to use a heuristic method to continue. For each independent submatrix $\mathbf{C}_i$ of $\mathbf{C}$ that has resisted the reduction rules and integer programming, we select one row containing a maximum number of 1-entries among the rows belonging to $\mathbf{C}_i$. The balls associated to these rows are marked as part of the solution and the rows are removed from $\mathbf{C}$. After such a selection in a greedy manner, the reduction rules from Subsection 3.6.1 are applied again, which, as observed for our data, again significantly reduces the number of 1-entries in $\mathbf{C}$. Afterwards, it is likely that new small independent parts do exist, and integer programming from Subsection 3.6.2 is applied to them again.

We repeat this process until $\mathbf{C}$ is empty, at which time $\mathrm{DMT}^*_{\mathrm{in}}$ is completed. That is, the union of the sets of points contained in the balls from $\mathrm{DMT}^*_{\mathrm{in}}$ is exactly S. Control flow of our hybrid algorithm of exact and heuristic methods is schematically depicted in Figure 3.2.

3.6.4 Overhead

Optimal solutions cannot be computed for large set covering instances, and the approximation guarantee of heuristics is unlikely to be improvable beyond a logarithmic factor in polynomial time [Fei96]. Therefore we want to determine the maximum overhead that is contained in the approximate solutions computed by our hybrid algorithm. One

way to calculate such a lower bound is to solve the linear program in Equation 3.5 under the relaxation $x_j \geq 0$. Clearly, this gives a lower bound of

$$|B^{\text{opt}}| \geq \lceil \sum x_j \rceil$$

for the optimal solution B^{opt}. However, even after redundancy removal (see Subsection 3.6.1) this method takes a considerable amount of time, and using the software *lpsolve* [LPS] we were not able to compute lower bounds for large inputs (see Section 3.9). We therefore give two alternative methods where bounds on the overhead are easy to obtain.

Lemma 3.7. *If the hybrid algorithm chooses g of the balls in DMT^*_{in} a greedy fashion (as in Listing 3.1), then*

$$|\text{DMT}^*_{\text{in}}| \leq B^{\text{opt}} + g.$$

Proof. After the last greedy pick, b_g, let B' be the set of balls still to choose from. Assume that the optimal solution for B' uses m balls. Taking no more greedy picks, our algorithm solves $B' \cup \{b_g\}$ with $m+1$ balls, whereas the optimal solution for $B' \cup \{b_g\}$ uses at least m balls. The lemma follows by induction. □

To get another bound, consider any set B of balls which cover S. For a ball $b \in B$ put $\alpha(b) = |b \cap S|$. Further, for a point $s \in S$, define its share as

$$share(s) = \frac{1}{\max_{s \in b \in B} \alpha(b)} .$$

Let $b(s)$ be a ball that achieves $\max_{s \in b \in B} \alpha(b)$. Under the (ideal) assumption that B is a disjoint covering of S (and hence covers each element s with ball $b(s)$ alone), $share(s)$ expresses the amount that s contributes to $|B|$. Thus, for any solution B, we have the lower bound

$$\lceil \sum_{s \in S} share(s) \rceil \leq |B|.$$

Let now B'' denote the set of balls being selected in the optimality keeping initial steps (i.e., before the first greedy pick). Clearly, the size of the optimal solution for $\text{DMT}^+_{\text{in}} \setminus B''$ differs from that for DMT^+_{in} by exactly $|B''|$. If we fix the shares for the elements in S with respect to the set $\text{DMT}^+_{\text{in}} \setminus B''$, then the following holds.

Theorem 3.8. *The overhead in the selected set DMT^*_{in} of balls is at most*

$$|\text{DMT}^*_{\text{in}}| - |B''| - \lceil \sum_{s \in S} share(s) \rceil.$$

3.6.5 Keeping the topology

In most cases, the unions $U(\mathrm{DMT}^*_{\mathrm{in}})$ and $U(\mathrm{DMT}^+_{\mathrm{in}})$ share the same topology, but there is no guarantee. In our experiments we have occasionally observed topological errors in $U(\mathrm{DMT}^*_{\mathrm{in}})$, especially when the sampling quality of S was poor.

Let $B_R = \mathrm{DMT}^+_{\mathrm{in}} \setminus \mathrm{DMT}^*_{\mathrm{in}}$ be the set of balls that have been removed during the pruning step. We can post-process $\mathrm{DMT}^*_{\mathrm{in}}$ with an algorithm that detects topological errors and repairs them by returning balls from B_R. The principal idea of our reparation approach is: If the topology of a union of balls changes due to removal of balls, then adding those balls to the pruned set of balls will change the topology of its union again. Five topological changes are conceivable when balls are added:

E1) An existing hole can disappear

E2) A new hole can appear

E3) A tunnel can disappear

E4) A new tunnel can appear

E5) Disconnected parts can get connected

Let $\mathcal{X}$ denote the set of all circular edges and vertices of the boundary of $U(\mathrm{DMT}^*_{\mathrm{in}})$. Each edge in $\mathcal{X}$ is defined by two intersecting balls from $\mathrm{DMT}^*_{\mathrm{in}}$. Likewise, each vertex in $\mathcal{X}$ is defined by the common intersection of (at least) three balls. We call an edge or vertex $x \in \mathcal{X}$ a critical intersection with respect to a certain ball b, if $x \cap b \neq \emptyset$ and the center of b is not contained in all balls by which x is defined. Let us assume that $b \in B_R$ is a ball which, when it is added to $\mathrm{DMT}^*_{\mathrm{in}}$ changes the topology of $U(\mathrm{DMT}^*_{\mathrm{in}})$. We treat all cases that can possibly occur (Note that our algorithm updates $\mathcal{X}$ upon insertion of balls, as later described in Listing 3.2):

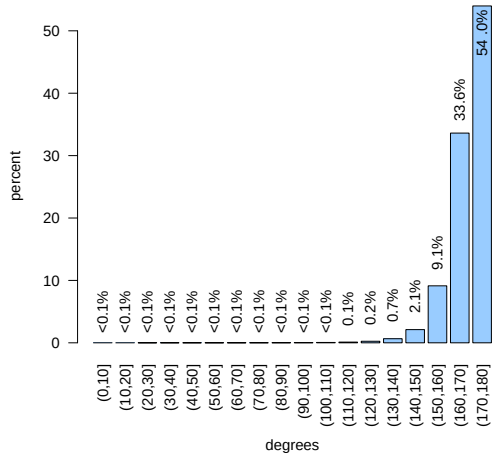
- ad E1: If b closes a hole, then b covers a vertex $x \in \mathcal{X}$ which is critical with respect to b, because the center of b can not lie within all balls whose boundaries form the hole.
- ad E2 and E3: If b generates a hole or interrupts a tunnel then b intersects a subset $R \in \mathrm{DMT}^*_{\mathrm{in}}$ whose union contains at least one tunnel, i.e. the balls in R have no common intersection. The consecutive intersections of the balls in R form circular arcs (edges) in $\mathcal{X}$ of which at least one is critical with respect to b.

```
Function TopologyRepair(DMT+in, DMT*in)
{
  B_R = DMT+in \ DMT*in
  B_T = DMT*in
  X = getBoundaryIntersections(B_T) // ...from the power diagram of B_T
  for each b ∈ B_R
  {
    if( (b has critical intersections in X) OR
        (insertion of b generates critical intersections))
    {
      DMT*in = DMT*in ∪ b
      return TopologyRepair(DMT+in, DMT*in) // restart the procedure
    }
    B_T = B_T ∪ b
    X = getBoundaryIntersections(B_T) // ...from the power diagram of B_T
  }
  return DMT*in
}
```

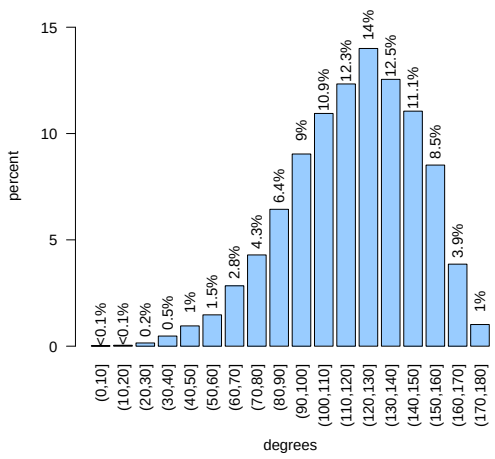
Listing 3.2: Topology verification and reparair

- ad E4 and E5: If inserting b reverses disconnectedness or generates new tunnels, then b connects two previously 'locally disjoint' components. Therefore, inserting b generates two ore more new elements in $\mathcal{X}$, at least one of which is critical with respect to b.

Listing 3.2 describes the algorithm to detect possible topological errors in $U(\mathrm{DMT}^*_{\mathrm{in}})$ and to repair them by returning balls. Note that this algorithm might yield a few false positives, i.e. it might return balls which are not essential for correct topology, but they do not harm, though. However, such false positives are unlikely because the balls under consideration intersect deeply. If two balls of the unpruned set $\mathrm{DMT}(\mathcal{O})$ define an edge on the boundary of the union $U(\mathrm{DMT}(\mathcal{O}))$, then they intersect at least at 120° as shown in Lemma 4.10. We have analyzed the intersection angles of the discrete medial axis transform of the *oilpump* model from Section 3.9. This model is not r-sampled, but the Lemma remained true for most of the angles, see Figure 3.3(a). For a heavily pruned version $\mathrm{DMT}^*_{\mathrm{in}}$ (where only every 220th ball was kept), the occurring intersection angles were still good, see Figure 3.3(b). We have not implemented this post-processing algorithm because—for reasonable sampling quality—the pruning works flawlessly without, as observed in our experiments.



(a) Oilpump, $|\mathrm{DMT}(\mathcal{O})| = 560928$ balls, min=4.51°, max=179.98°, avg=168.67°



(b) Oilpump, $|\mathrm{DMT}^*_{\mathrm{in}}| = 2546$ balls, min=7.40°, max=178.88°, avg=118.03°

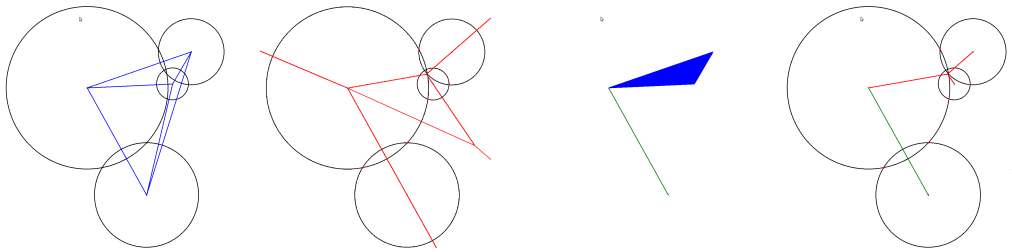
Figure 3.3: Histograms of ball intersection angels at the boundary of a union of balls as observed in our experiments with the *oilpump* model from Section 3.9.

3.7 Computing the medial axis

After generating, enlarging, pruning, and post-processing the set of balls in order to get a suitable discrete medial axis transform $\mathrm{DMT}_{\mathrm{fin}}$ of the input object $\mathcal{O}$, it just remains to compute the medial axis of the union of the balls in $\mathrm{DMT}_{\mathrm{fin}}$. This medial axis is guaranteed to be homotopy equivalent to $\mathcal{O}$. Moreover, it has a particularly beneficial structure, as it consists of piecewise linear primitives, which can be computed exactly. An algorithm for doing so has been described in [AM97, AK01]. We refrain from repeating the details in these papers and only give a short overview.

- Compute the dual graph of $\mathcal{PD}(\mathrm{DMT}_{\mathrm{fin}})$, which is a regular triangulation RT; see Figure 3.4(a).
- Consider the union of the balls in $\mathrm{DMT}_{\mathrm{fin}}$, and compute the Voronoi diagram $\mathcal{V}$ of the vertices of this union; see Figure 3.4(b).
- Extract the weighted zero alpha shape $\mathcal{A}$ of $\mathrm{DMT}_{\mathrm{fin}}$, using $\mathcal{A} \subseteq RT$. Split $\mathcal{A}$ into its singular (not full-dimensional) part S and its regular part R; see Figure 3.4(c).
- Output the medial axis of $\mathrm{DMT}_{\mathrm{fin}}$, which is $S \cup (R \cap \mathcal{V})$; see Figure 3.4(d).

We have implemented this algorithm. Examples and running times are presented in Section 3.9.



(a) A regular triangulation on top the weighted points (balls) in $\mathrm{DMT}_{\mathrm{fin}}$

(b) Voronoi diagram $\mathcal{V}$ of the vertices of the union of $\mathrm{DMT}_{\mathrm{fin}}$

(c) Weighted zero alpha shape $\mathcal{A}$, consisting of a regular part R (blue) and a singular part S (green)

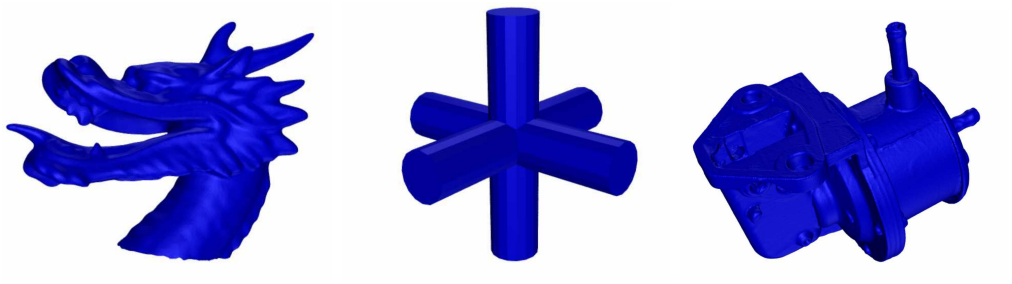
(d) The medial axis of $\mathrm{DMT}_{\mathrm{fin}}$, consisting of $S \cup (R \cap \mathcal{V})$

Figure 3.4: Medial axis algorithm

3.8 Implementation details

3.8.1 Matrix storage model

Let $k = |\mathrm{DMT}^{+}_{\mathrm{in}}|$ and $n = |S|$. By construction, each Voronoi vertex can be a pole point for up to four sites from S. Therefore, $k \leq n$ holds, although in practice k is always in the range of n, as observed in our computations. For practical purposes, it



(a) Dragon, 71720 triangles

(b) 3Cross, 107136 triangles

(c) Oilpump, 1140049 and 350000 triangles, respectively

Figure 3.5: The three input models

is convenient to regard the $(k \times n)$ covering matrix $\mathbf{C}$, introduced in Section 3.6.1, as a $(n \times n)$ matrix, and we simulate the $n-k$ rows by duplicating the balls which belong to more than one sample point. This way $\mathbf{C}$ becomes a quadratic, non-symmetric binary matrix. A fundamental question of the set covering implementation was to choose the right storage model for $\mathbf{C}$. We had to account for the following requirements:

1. $\mathbf{C}$ can get huge, i.e. it can consist of a million lines and rows but it is typically only sparsely populated by non-zero elements, so a sparse storage model is required.

2. The row reduction rule of our approach frequently needs to check if a row of $\mathbf{C}$ is a subset of another row, so fast access to only its non-zero elements is required.

3. The column reduction rule frequently needs to check if a column of $\mathbf{C}$ is a superset of another column, which requires fast access to only its non-zero elements.

No program library supported all requirements, so we had to implement our own matrix representation class. In the following, we refer by the term *STL-set* to the container *set* of the C++ Standard Template Library. This container stores its contents in a balanced binary tree, such that the time consumption for the operations *find* and *remove* is logarithmic in the container size.

We store the rows of $\mathbf{C}$ in a collection of *STL-sets* as follows: For $i = 1, \cdots, n$ the *STL-set* R_i stores the indices of all columns of $\mathbf{C}$, that contain a non-zero entry in the ith row. By its own, this scheme accounts for the requirements 1 and 2, but it leaves requirement 3 unconsidered. Hence, to answer the question if column j_1 is a superset of column j_2 we would need to check for all *STL-sets* $\{R_1, \cdots, R_n\}$ if the value j_2 is contained, and if so, check them also for j_1. To meet also requirement 3, we store another copy of $\mathbf{C}$, namely using another collection of *STL-sets* $\{C_1, \cdots, C_n\}$, where the C_j stores those rows of $\mathbf{C}$ that contain a non-zero entry in the jth column. This storage scheme fulfills the described requirements, at the price of double memory consumption and the requirement to perform update operations on $\mathbf{C}$ twice.

3.8.2 Fast subset tests

The row reduction rule of our approach needs to determine all rows of $\mathbf{C}$ being a subset of any other row. This is similarly true for the column reduction rules, where all supersets need to be determined. We stick in our explanations with the row reduction rule as for columns things work analogously.

A naive implementation would compare all n^2 pairs of rows with each other, considering every element contained within. Although our storage model is very advantageous for these tests, such an implementation would be way too expensive when n gets in the range of a million or even only a tenth of it. We had to find a clever way to speed up these tests significantly, and we have implemented two different approaches. One approach involves hash values that eliminate the need to compare a pair of rows by their non-zeroes in many cases. The other one computes a permutation of $\mathbf{C}$ to minimize the number of pairs to be compared.

The hashing approach

Our idea to speed up the individual row-to-row comparisons was to eliminate the need to struggle with their data elements. To this end we augment $\mathbf{C}$ with hash values. More accurately, we add to each row R_i of $\mathbf{C}$ a 32-bit hash value r_i that represents the information contained in R_i in a compressed and simplified form. Let us denote the k-th bit of r_i by $r_i(k)$, and let t denote a certain threshold that can be chosen appropriately. We assign $r_i(k)$ as follows:

$$r_i(k) = \begin{cases} 1 & \sum_{j=k\cdot\frac{n}{32}}^{(k+1)\cdot\frac{n}{32}-1} x_{i,j} > t \\ 0 & else \end{cases}$$

This scheme virtually splits R_i into 32 sections. When the number of non-zero elements exceeds t in the k-th section, then $r_i(k)$ is set, otherwise it is cleared. Now, when we test if R_{i2} is a subset of R_{i1}, we first evaluate

$$r_{i1} \text{ OR } r_{i2} \overset{?}{>} r_{i1} \tag{3.6}$$

If the predicate in Equation 3.6 yields true, then there is at least one k, for which $r_{i2}(k) > r_{i1}(k)$ holds, and then the k-th virtual section of R_{i2} contains more non-zeroes than the k-th virtual section of R_{i1}. Consequently, R_{i2} cannot be a subset of R_{i1}, and we are done. Else, if the predicate yields false, we must compare R_{i1} and R_{i2} by their data elements.

Every CPU in a personal computer supports the bitwise OR operation of Equation 3.6 in hardware. Thus we can compare all 32 bits in a computationally cheap way in one task. We have implemented this approach using 32 counters per row which are updated whenever their corresponding row of $\mathbf{C}$ is updated. This approach works

quite well, and compared to the naive implementation, it reduces the running time considerably.

Bandwith reduction of C

The approach described in Section 3.6 saves time. But still, every time the row reduction rule is applied to $\mathbf{C}$, $O(n^2)$ operations have to be performed. Fortunately, I was on a research stay at Inria, Sophia Antipolis, back in 2006, where I gave a talk and mentioned the problem. After my talk, Pierre Alliez, a researcher in the Geometrica Group, proposed to compute a bandwidth reducing permutation of $\mathbf{C}$ to eliminate the need to compare $O(n^2)$ pairs of rows. The very successful approach that we derived from Pierre's idea is described in the following.

Definition 3.9. *Let $G = (V, E)$ denote an undirected, connected graph, where V is the set of vertices and E is the set of edges, and let $A(G) = \{a_{i,j}\}$ denote the adjacency matrix of G. The bandwidth β of $A(G)$ is defined as*

$$\beta = \underset{a_{i,j} \neq 0}{max} |i - j|$$

Generally speaking, if $A(G)$ is sparse, it is for many problems beneficial to minimize its bandwidth before further operations are performed with $A(G)$. To this end, the vertices in $G = (V, E)$ must be re-labeled, such that all non-zeroes of $A(G)$ get as close as possible to its main diagonal. This is the so called *bandwidth minimization problem*, known to be NP-complete [Pap76], and the classical *Cuthill-McKee* [CM69] algorithm is an approach for approximate results, i.e., bandwidth reduced matrices.

In our case, reducing the bandwidth of the covering matrix $\mathbf{C}$ is also very beneficial, as will be described at the end of this section. But $\mathbf{C}$ is neither symmetric nor necessarily connected, so we have devised a variant of the Cuthill-McKee algorithm, that accounts for these specifics.

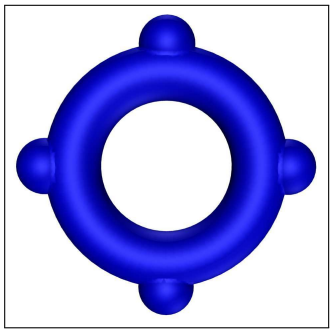
First we (virtually) derive from $\mathbf{C}$ a new $n \times 2n$-matrix $\mathcal{Y}$, whose elements are

$$y_{i,j} = \begin{cases} x_{i,j-n} & j > n \\ 0 & else \end{cases}$$

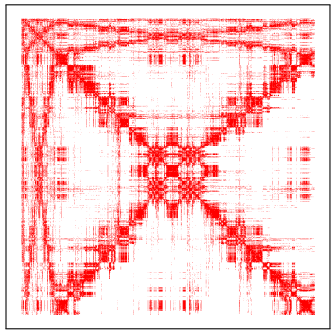
The new matrix $\mathcal{Y}$ describes a directed graph G_y on top of $2n$ vertices. We use $\mathcal{Y}$ as input for our Cuthill-McKee-variant, which can be thought of as a classical breadth first search in G_y. As a specialty of this breadth-first search the neighbors of a certain

```
function cuthillMcKeeVariant(𝒴)
Vector R,C; // Output of the algorithm
Queue Q; // Queue for breadth-first search
while(|R| < n and |C| < n)
{
  Q.push_back(any unexplored vertex with min. outdegree); // Init queue
  while(Q.size() > 0) // Until connected component is fully explored...
  {
    Vertex v=Q.front(); // Get first vertex v from Q
    Q.pop_front(); // Remove v from Q
    if(v ≤ n)
    {
      R.push_back(v);
      Vector NEIG={j|y_{v,j} ≠ 0, j ∉ (C ∪ Q), j = n+1, ··· , 2n};
      sort_vertices_ascendingly_by_outdegree(NEIG);
      Q.push_back(NEIG);
    }
    else
    {
      C.push_back(v);
      Vector NEIG={i|y_{i,v} ≠ 0, i ∉ (R ∪ Q), i = 1, ··· , n};
      sort_vertices_ascendingly_by_outdegree(NEIG);
      Q.push_back(NEIG);
    }
  }
}
return (R,C);
```

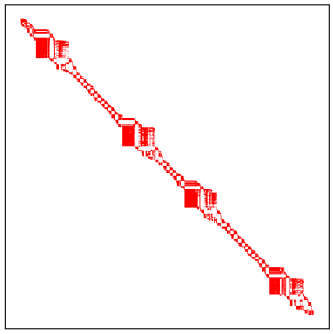
Listing 3.3: Computing the permutation vectors R and C



(a) The *torusBalls* model



(b) Scatter plot of the original covering matrix **C**. Rows correspond to balls, columns correspond to sample points of *torusBalls*



(c) The bandwidth limited version of **C**. Rows and columns have been re-labeled

Figure 3.6: The *torusBalls* model and scatter plots of the non-zero elements of the corresponding covering matrix **C**.

vertex v are sorted by their outdegree before they are inserted into the queue Q as described in Listing 3.3.

The output of this algorithm are two vectors R and C, and we use them to compute a permutation of $\mathcal{Y}$. Thereby, the number r on the i-th position of R means that the r-th row of $\mathcal{Y}$ has to be given the new row number i. For C and the columns of $\mathcal{Y}$ this is done in the same manner. After the permutation, the first n rows and columns of $\mathcal{Y}$ contain the desired result, i.e., the bandwidth-limited version of **C**.

As to our implementation: In fact, the second matrix $\mathcal{Y}$ serves only as a means to describe the algorithm. In practice, our implementation operates only on **C** and it simulates the existence of $\mathcal{Y}$ by index shifting. We illustrate the results of our algorithm using the *torusBalls* model, shown in Figure 3.6(a) as input object $\mathcal{O}$. We have computed its discrete medial axis transform DMT($\mathcal{O}$) and have enlarged the radii of these balls by 0.001 times the length of the largest edge of the object's axis aligned bounding box. The corresponding covering matrix **C** is shown in Figure 3.6(b). We have applied our Cuthill-McKee-variant, and the resulting covering matrix **C** is shown in Figure 3.6(c).

Now, what is the big advantage of the bandwidth-reduced version of **C**? Let $f_{i,j}$ denote the first non-zero element of a certain row i of **C**. The described breadth-first algorithm visits each vertex in G_y exactly once. Further, it inserts the vertices into the

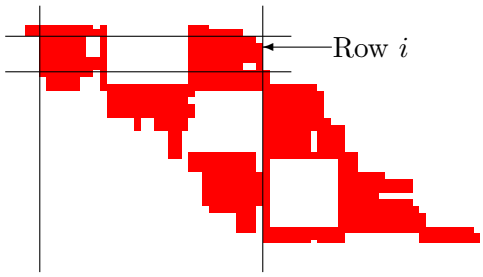


Figure 3.7: Magnification of Figure 3.6(c). Rows with entries outside the two vertical lines cannot be a subset of the i-th row.

queue Q and later into R or C in the order they are explored. As a consequence, when we consider $f_{i,j}$ and $f_{i',j'}$ with $i' > i$, then $j' \geq j$ holds, see Figure 3.7. This property greatly limits the range of candidate rows, possibly being a subset of a certain row i.

3.9 Examples and evaluation

In this section, we explore the behavior of our implementations with respect to running time and memory consumption. Three input models with quite different properties are considered. As a widely known and commonly used model we have chosen *dragon*, see Figure 3.5(a), which is a part of the dragon model from the Stanford 3D Scanning Repository [STA]. We have constructed the *3Cross* model in Figure 3.5(b), whose parts of the medial axis are easy to imagine. Finally, we use the original as well as a down-sampled version of the *oilpump* model from the Aim@Shape repository [AIM] as technical model with many sharp edges, see Figure 3.5(c). Though these models are not necessarily r-sampled, our algorithm works well in these realistic situations.

The software has been implemented as far as useful for scientific purposes. Local feature size computations, although easy to implement after our theoretic work, are not included. The models can nevertheless be rendered because only tiny values are required for the ball enlargement parameter ε to greatly reduce and stabilize the balls from $\mathrm{DMT}_{\mathrm{in}}$.

Our applications have been compiled with gcc [GCC] for 64bit Linux systems, and "O3" optimization was set. CGAL 3.3.1, on which our applications are based, is not thread-safe, so we have completely renounced parallel computing, and all applications are single-threaded. As large models may require lots of memory, we have carried out all benchmarks on server hardware, namely on an Intel XEON E5345 CPU running at 2.33 GHz, where 16 GB RAM were available.

3.9.1 Discrete Medial Axis Transform

Table 3.1 shows the time and memory consumption of our discrete medial axis transform software, that takes a triangulated boundary representation of the object as input and computes the set $\mathrm{DMT}_{\mathrm{in}}$. The main time consumption is caused by the part which extracts pole points from the Voronoi diagram. This includes distance computations, sorting of distances, as well as breadth-first searches on the input surface mesh for labeling the Voronoi vertices as being inside or outside the object. The latter is more expensive for dense meshes, as can be seen from the growth of the running time for the oilpump model, which has been rendered in two different resolutions. The memory consumption behaves linearly, as one may expect.

	Input	Output	Time	Memory
	Vertices	$\lvert\mathrm{DMT}_{\mathrm{in}}\rvert$	[sec]	[MB]
Dragon	35862	35676	74.76	367.27
3Cross	53570	52946	289.74	622.55
Oilpump small	174994	170106	1287.32	1798.01
Oilpump large	570018	560928	9070.44	5908.74

Table 3.1: Discrete medial axis transform

3.9.2 Covering matrix

The time requirement for building the covering matrix consists of the time needed to construct a kd-tree of the sample points S, and of the time needed for the spherical range searches. The time consumption for these queries is output sensitive. It depends on the ball enlargement parameter ε as well as on the shape of the object.

Table 3.2 displays the time for calculating the covering matrices, where we have enlarged the balls additively by certain percentages of the longest edge of the object's bounding box (shown as labels heading the columns in Table 3.2). We keep the memory consumption of this part of our toolchain low by storing only the kd-tree in main memory, while the output of the queries is immediately written to a compressed binary file.

3.9.3 Set covering

Our set covering software takes the covering matrix from Section 3.9.2 as input, where the covering information of $\mathrm{DMT}_{\mathrm{in}}^{+}$ with respect to the original sample points is encoded,

	$\lvert DMT^+_{in}\rvert$	0.01%	0.05%	0.10%	0.30%	0.50%
		[sec]	[sec]	[sec]	[sec]	[sec]
Dragon	35676	44.1	46.7	50.0	62.2	72.8
3Cross	52946	209.3	220.1	230.5	262.4	284.8
Oilpump small	170106	417.4	580.8	751.9	1272.9	1805.0
Oilpump large	560928	2475.5	3668.9	4591.3	7858.1	11649.9

Table 3.2: Covering matrix

and it determines a small subset $DMT^*_{in} \subset DMT^+_{in}$ that still covers all the sample points. In Table 3.3 we compare our hybrid algorithm with the simple greedy heuristic. In column 'Over' we show the upper bound on the overhead in balls for these instances, according to the formulas in Subsection 3.6.4. It becomes apparent that almost minimal sets of balls are computed. Our application allows to choose between different trade-offs regarding running time, memory consumption, and quality of the result, and the benchmarks have been performed such that high quality results were computed. At entries '•' the tasks were too memory consuming for our hardware. We could still solve such huge instances by reducing the amount of data in a preprocessing step, but the results would not be comparable, so we have refrained from doing so.

	Input #balls	ε	Greedy #balls	Hybrid #balls	Over #balls	Hybrid [s]	Hybrid [MB]
Dragon	35676	0.01	6883	6035	120	43.8	38
	35676	0.05	3709	3212	276	85	61.6
	35676	0.10	2422	2062	327	91.2	92.2
	35676	0.30	1022	842	181	120.1	173.3
	35676	0.50	628	508	114	101.4	242.7
3cross	52946	0.01	4874	4252	164	96.9	118.8
	52946	0.05	2847	2529	117	252.4	151
	52946	0.10	2151	1848	121	56	355
	52946	0.30	465	387	58	59.9	564.6
	52946	0.50	329	253	39	91.4	748.7
Oilpump	170106	0.01	12608	10883	1632	2868.3	388.9
	170106	0.05	3027	2546	871	4165	1502.9
	170106	0.10	1386	1163	416	4864.6	2890.7
	170106	0.30	285	231	91	10418.7	9581
	170106	0.50	•	•	•	•	•

Table 3.3: Set covering: Greedy and hybrid solutions

3.9.4 Medial axis computation

This part of our toolchain is a straight implementation of the approach described in [AK01]. Table 3.4 shows the running time and memory usage of the medial axis application. Its time consumption depends not only on the number of input balls but is also affected by the complexity of the connected components of the respective weighted alpha shape. For example, the more pruned versions of the *3cross* model contain only few regular components, and their per-ball time consumption is less than a third of one of the *dragon* model, which comprises large connected components.

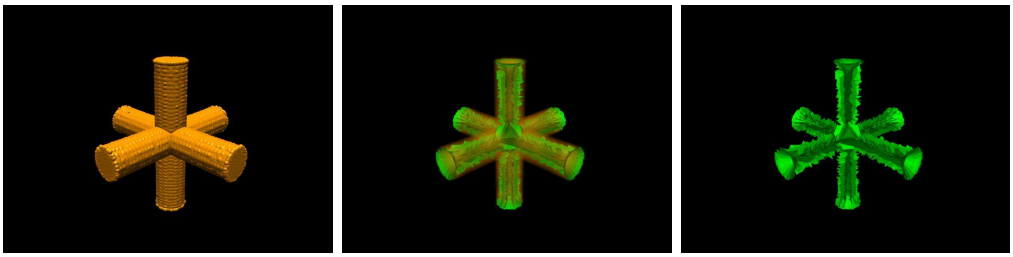
In our first example, the input object $\mathcal{O}$ is the *3cross* model, and we have computed approximations at two different degrees of pruning. Figures 3.8(a) to 3.8(c) show a version, where $\mathrm{DMT}^*_{\mathrm{in}}$ is insufficiently pruned, and thus the corresponding discrete medial axis $\mathrm{DM}(\mathcal{O})$ differs significantly from an intuitive medial axis of the apparently smooth *3cross* model. One might conclude that the unwanted branches are due to inaccuracies of the Voronoi approach (see Section 3.4). However, the contrary is true: $\mathrm{DMT}_{\mathrm{in}}$ is a good approximation of $\mathrm{MT}(\mathcal{O})$, and therefore accurately models the unstable behavior of $\mathrm{M}(\mathcal{O})$ with respect to perturbations on $\mathcal{F}$. Such perturbations indeed exist in the *3cross* model, but are due to a poor sampling strategy of the used modeling software; cf. Figure 3.9. No matter whether perturbations are due to poor sampling or noise, the artifacts disappear after sufficient pruning, as is shown in Figures 3.8(d) to 3.8(f).

In Figure 3.10, we display the *dragon* model at two different levels of pruning. Its medial axis has a particularly complicated structure, and our approach proved to be capable of dealing with all the small details in a topologically correct way. This shows the flexibility of our method, in the sense of adapting itself to coarser and finer object details.

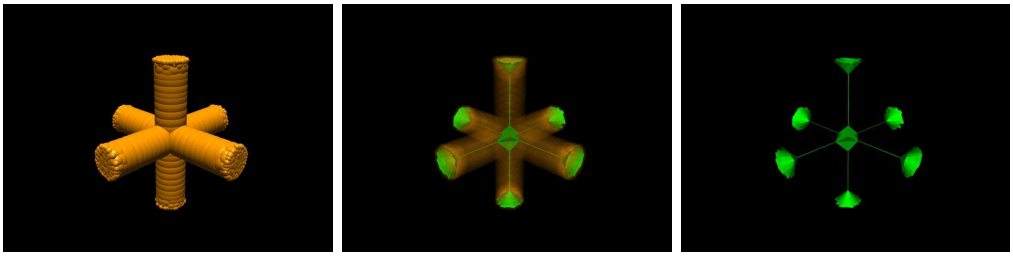
The largest model in our experiments is the *oilpump* model. This model features many details and sharp edges. The corresponding point sample does therefore certainly not keep the r-sampling condition, and the original Voronoi approach would fail. With our extensions with respect to the labeling strategy, see [AAH+07], the model can nevertheless be rendered. The time and memory consumption (as shown in Table 3.4) for more than 10 000 balls indicates that there is still potential for even larger, more detailed inputs.

	Balls	[sec]	[MB]
Dragon	6035	192.25	264
Dragon	3212	72.24	119.9
Dragon	2062	38.52	70.38
Dragon	842	15.34	28.82
Dragon	508	7.88	16.3
3Cross	4252	287.44	217.17
3Cross	2529	113.34	124.09
3Cross	1848	106.84	85.43
3Cross	387	1.64	10
3Cross	253	1.17	6.99
Oilpump	10883	105.78	378.15
Oilpump	2546	17.8	80.18
Oilpump	1163	6.42	37.78
Oilpump	231	0.92	9.91

Table 3.4: Medial axis computation



(a) 2529 Balls (b) Transparent (c) Medial axis



(d) 387 Balls (e) Transparent (f) Medial axis

Figure 3.8: The 3cross model

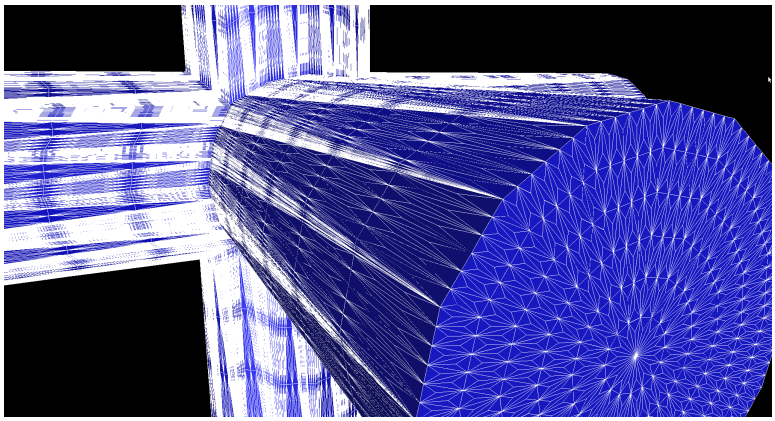
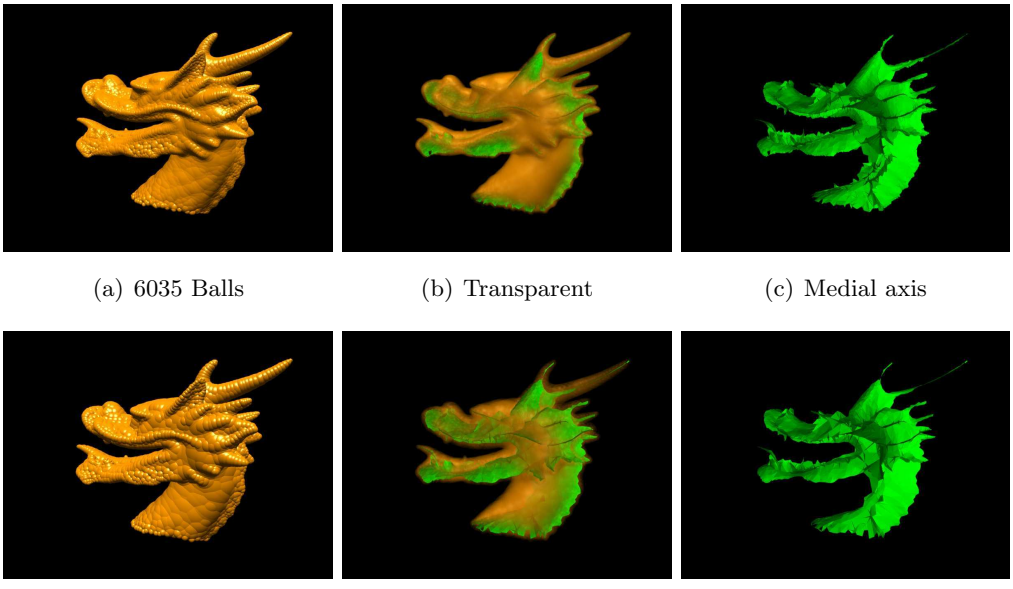


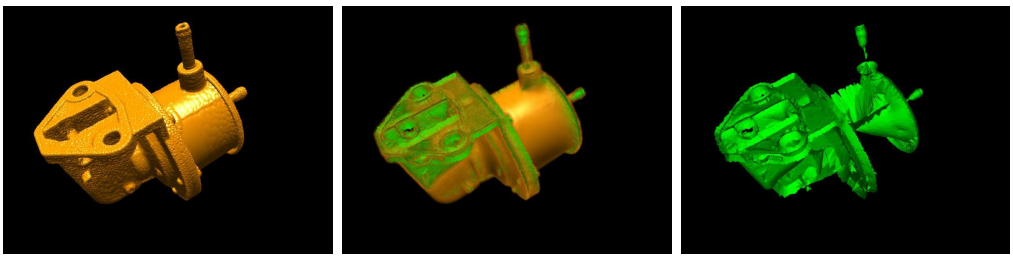
Figure 3.9: Bad sampling of the *3cross* model



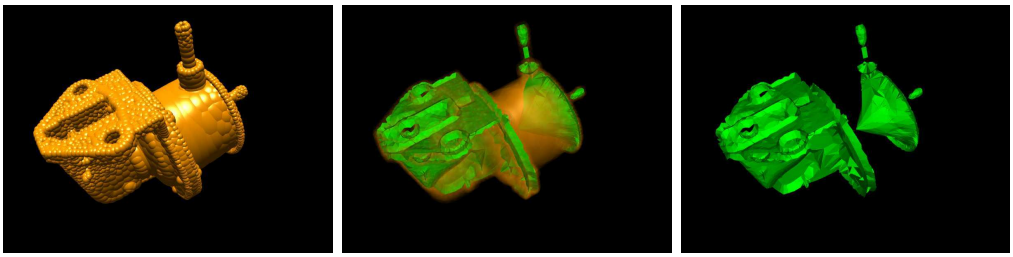
(a) 6035 Balls (b) Transparent (c) Medial axis

(d) 2062 Balls (e) Transparent (f) Medial axis

Figure 3.10: The dragon model



(a) 10883 Balls (b) Transparent (c) Medial axis



(d) 2546 Balls (e) Transparent (f) Medial axis

Figure 3.11: The oilpump model

3.10 Conclusions

We have developed a toolchain, which takes boundary triangulated 3D objects as input, computes approximations by balls, stabilizes and prunes them, and extracts the medial axes of their union, as approximate medial axes of the original objects. To this end, we have significantly modified and extended the technique described in [AB99, AK00] to robustly approximate objects by unions of balls. We have proven the topological correctness of our constructions, which relies heavily on r-sampling theory. The stability problem, inherent to medial axis constructions, is coped with set covering. To the best of our knowledge, pruning medial axes through set covering is a novel approach. Considerable effort has been spent to implement a set covering application which combines exact and and heuristic methods, and it does in practice not only achieve better results than a simple greedy algorithm, but can also bound the overhead with respect to the optimal solution. To be able to evaluate our constructions, we have also implemented the medial axis algorithm in [AK01]. We are aware of the fact that it is not realistic to expect r-sampled input in a real world setting. Our software works well even if the input objects fail to be r-sampled within certain ranges.

Chapter 4

Surface Reconstruction and Seed Polytopes

4.1 Abstract

For a surface $\mathcal{F}$ in 3-space that is represented by a set S of sample points, we construct a coarse approximating polytope P that uses a subset of S as its vertices and preserves the topology of $\mathcal{F}$. Thereby, we use as few points from S as possible. Such a polytope P is useful as a 'seed polytope' for starting an incremental refinement procedure to generate better and better approximations of $\mathcal{F}$ based on interpolating subdivision surfaces or e.g. Bézier patches.

Our algorithm starts from an r-sample S of $\mathcal{F}$ and constructs a set of surface balls, whose centers are the points in S. The radii are carefully chosen, such that the topology is retained. From the weighted zero α-shape of a proper subset of these highly overlapping surface balls we get the desired polytope. By the choice of the radii of the surface balls one can control the degree by which sample points are omitted. Thus, our method can also be used for coarse-to-fine surface reconstruction.

4.2 Introduction

This chapter of the thesis deals with recovering structural information for a 3-dimensional object that is represented by a sample point cloud. More specifically, given an object $\mathcal{O}$ in 3-space and an r-sample S of its boundary, we want to find an approximating polytope P that uses a subset of the points in S as its vertices and preserves the topology of $\mathcal{O}$. Our goal is, on the one hand, to use as few points of S as possible and, on the other, to get a flexible approximation whose level of detail can be tuned from coarse to fine. Motivation for studying this problem is based on open problems in object simplification and surface reconstruction, two fundamental tasks in several areas of computer science, like geometric modeling, computer graphics, and computational geometry.

The main support structure we use is an approximation of the object in question with a union of balls. In the context of object simplification, this approach is used for many purposes, e.g. collision detection [Hub96], shape matching [SS04], and shape interpolation [RF96], to name a few. Regarding surface reconstruction, approximating objects with balls also plays a major role, see for example the power crust algorithm [ACK01], related work [AB99, AK00, AK01] and also [CL08], naming again only a few.

In our approach, which is similar to work in [CL08], we build a union of so-called *surface balls*, centered at the points in our r-sample S on the surface $\mathcal{F}$ of $\mathcal{O}$, whose radii adapt to the local feature size of $\mathcal{F}$. The desired approximating polytope P is then extracted from the weighted α-shape [Ede95] (for $\alpha = 0$, of a carefully chosen subset of these balls. In contrast to [CL08], where prior knowledge of the local feature size of $\mathcal{F}$ is assumed, we obtain an estimation of this function from the data, by using distances to poles, see Definition 3.2. Using a tailored technique of pruning the surface balls, we obtain a coarse-to-fine approximation of $\mathcal{F}$ by polytopes. This is the first result that uses, from a practical point of view, approximations of local feature size and medial axis to obtain locally adaptive reconstructions of an unknown surface.

The polytopes we construct are topologically correct reconstructions of $\mathcal{F}$. Thus our results differ from existing multi-scale surface reconstruction techniques in [NSW08, CL08, CCSL09, GO08] where topological filtering occurs. At the coarsest level, the polytope we obtain is what we call a 'seed polytope', as it provides not only a coarse approximation of $\mathcal{F}$ but also a mapping of the non-used sample points in S to the vertices of the polytope. Such a mapping is needed for incrementally generating approximations of $\mathcal{F}$ based on interpolating subdivision surfaces or Bézier patches. We

stress that the intended purpose of the seed polytope is not primarily in approximating $\mathcal{F}$ but rather in serving as a (topologically correct and small) starting structure for subsequent approximations by patches. We thus do not try to keep the approximation error small for the seed polytope itself, and use this additional freedom to keep the polytope small.

4.3 History and contributions

This chapter describes joint work with my co-authors Astrid Sturm and Günter Rote from the Institute of Computer Science, Free University Berlin, Simon Plantinga and Gert Vegter from the department of Mathematics and Computer Science, University Groningen, as well as Franz Aurenhammer from the Institute for Theoretical Computer Science and Oswin Aichholzer from the Institute for Software Technology, both Institutes belonging to Graz University of Technology.

Previously to this work, the two papers *Convex approximation by spherical patches*, [BPR+07], and *Approximating boundary-triangulated objects with balls*, [AAH+07], have evolved independently from each other. The ultimate goal of our joint work was to combine the previous work into a new technology to approximate non-convex objects by curved patches. The outcome of our research has been published in the paper *Recovering Structure from r-sampled Objects*, [AAK+09], as well as in [Stu09].

4.4 Definitions and notation

We continue to use the notation and definitions which we have introduced in Section 3.3 and 3.4. In particular, we denote the object we want to reconstruct by $\mathcal{O}$ and its surface by $\mathcal{F}$, and we assume that $S \in \mathcal{F}$ is an r-sample of $\mathcal{O}$ for $r \leq 0.08$.

Definition 4.1.

- *The* discrete medial axis $\mathrm{DM}_{\mathrm{in}}$ ($\mathrm{DM}_{\mathrm{out}}$) *is the medial axis of the union of polar balls in the sets* $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$ ($\mathrm{DMT}(\mathcal{O})_{\mathrm{out}}$).
- *The* discrete local feature size $\widetilde{\mathrm{lfs}}(x)$ *of a point $x \in \mathcal{F}$ is the minimum distance from x to* $\mathrm{DM}_{\mathrm{in}} \cup \mathrm{DM}_{\mathrm{out}}$.
- *The* pole distance $\hat{D}(x)$ *of a point x is the distance to the nearest pole.*

We will see that $\hat{D}$ is a good estimate of $\widetilde{\text{lfs}}$ (Corollary 4.12), as well as an upper bound on the true local feature size (Lemma 4.8). In practice, $\hat{D}$ is easier to compute than $\widetilde{\text{lfs}}$, and the true local feature size is not computable at all.

The *weighted α-shape* for $\alpha = 0$ is the dual shape of a union of balls [Ede95], see the definition in Section 2.6. It is a simplicial complex whose vertices are the centers of the balls, and which is homotopy-equivalent to the union of balls. We will refer to the weighted zero α-shape of $\text{DMT}(\mathcal{O})_{\text{in}}$ as $\mathcal{A}_{\text{in}}$ and to the one of $\text{DMT}(\mathcal{O})_{\text{out}}$ as $\mathcal{A}_{\text{out}}$.

Proposition 4.2. *[AK01] Let $\mathcal{A}_{\text{in}}$ and $\mathcal{A}_{\text{out}}$ be the weighted zero α-shapes of $\text{DMT}(\mathcal{O})_{\text{in}}$ and $\text{DMT}(\mathcal{O})_{\text{out}}$. Then we have*

$$\text{DM}_{\text{in}} \cup \text{DM}_{\text{out}} \subseteq \mathcal{A}_{\text{in}} \cup \mathcal{A}_{\text{out}}.$$

4.5 Our approach

4.5.1 The union of surface balls.

A *surface ball* is a ball with center at a sample point $s \in S$. For seed polytopes, our goal is to represent the surface $\mathcal{F}$ of $\mathcal{O}$ in a topologically correct way with as few faces as possible. We try to make the surface balls as large as possible, while guaranteeing correct topology of the the union $U(B_{\mathcal{F}})$ of the set $B_{\mathcal{F}}$ of surface balls. A subsequent pruning step will delete some of these balls whenever the sample is denser than necessary. For surface reconstruction, we will output meshes of adjustable complexity. The only modification necessary to reach this goal is to choose surface balls with smaller radii.

4.5.2 Pruning

We prune $B_{\mathcal{F}}$ with the set covering approach that we have already used for discrete medial axis transforms in Section 3.6. This approach computes a small subset $B'_{\mathcal{F}}$, in a purely combinatorial manner, without regard to geometry and topology, i.e. it assures that S is covered by the union of $B'_{\mathcal{F}}$, not necessarily $\mathcal{F}$. Thus, we needed to find a way to keep the set covering step from punching holes in our constructions.

Before we launch the set covering procedure, we (virtually) shrink the balls in $B_{\mathcal{F}}$. This is done such that covering of S by a subset of shrunk balls guarantees that the corresponding original surface balls cover the surface $\mathcal{F}$, and moreover, their union represents the topology of $\mathcal{F}$ correctly.

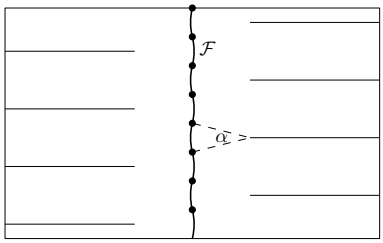


Figure 4.1: A wiggly curve $\mathcal{F}$ with a point sample on a straight line. (Adapted from [AK00].)

4.5.3 The polyhedral approximation.

Finally we compute the weighted zero α-shape of $B'_{\mathcal{F}}$, which has the same topology as $\mathcal{F}$ and which gives the desired seed polytope. The vertices of the weighted zero α-shape are points in S, because the centers of the balls in $B'_{\mathcal{F}}$ have been chosen from S. We use the power diagram of $B'_{\mathcal{F}}$ to find out which vertex of the polytope each sample point $s \in S$ belongs to, and provide a list of pointers representing this relation.

4.5.4 Obtaining the local feature size.

A distinguishing feature of our problem setting is that we cannot get a lower estimate on the local feature size. Figure 4.1 shows a section of a curve $\mathcal{F}$ that consists of alternating short circular arcs. The horizontal lines are part of the medial axis. The points of the r-sample S are aligned vertically. By reducing the angle α, such an example can be built for any $r > 0$. The algorithm sees only these samples. Thus, to the algorithm, this input is indistinguishable from a very densely oversampled straight line.

4.6 Technical results

In order to generate adequate sets of polar balls and surface balls (in both cases, the topology must be maintained), we need to derive certain information concerning the local feature size of the sampled object. The present and the subsequent section are devoted to this issue. We obtain several new properties of r-sampled objects for suitable values of r.

Let M_{in} and $\mathrm{M}_{\mathrm{out}}$ denote the inner and the outer (real) medial axis of the given object $\mathcal{O}$, respectively. We start by bounding the distance of poles to the respective

parts of the medial axis—a result crucial for bounding the radii of surface balls in Section 4.7.

Theorem 4.3. *For an r-sample S, let p be an inner (resp. outer) pole of a sample point $s \in S$, and denote with B_p the inner (outer) polar ball of s, with radius R_p. The distance from p to* M_{in} ($\mathrm{M}_{\mathrm{out}}$) *is at most $O(r) \cdot R_p$.*

In the limit, when the sampling density approaches zero, poles and the medial axis coincide, as has already been shown by Amenta et al. [ACK01, Theorem 35]. In contrast to this result, we give an explicit quantitative analysis in terms of r. Results similar to Theorem 4.3 have been shown (see e. g. [ACK01, Lemma 34], on which Theorem 35 is based, or [BC01, Proposition 16]). However, we could not use these results, since they hold only when the angle between the two closest surface points to a given point on M_{in} ($\mathrm{M}_{\mathrm{out}}$) is not too small, (These points form the *γ-medial axis*.)

Proof. The idea of the proof is to turn the polar ball B_p into a medial ball, while not moving its center too much. The proof is based on several technical lemmas which are given subsequently. We proceed in three steps, see Figure 4.2(a):

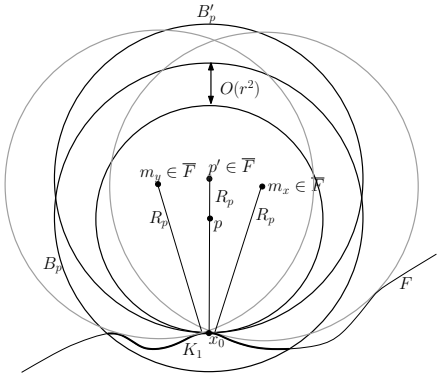
1. While keeping the center of B_p fixed we shrink the radius of B_p until the ball becomes empty, touching the surface $\mathcal{F}$ of $\mathcal{O}$ at some point x_0. By Lemma 4.4 below, the difference Δ_1 between the new radius and the original radius R_p is at most $\Delta_1 = O(r^2) \cdot R_p$.

2. We expand the shrunken ball from the touching point x_0 by moving its center in the direction $\overrightarrow{x_0 p}$ until either

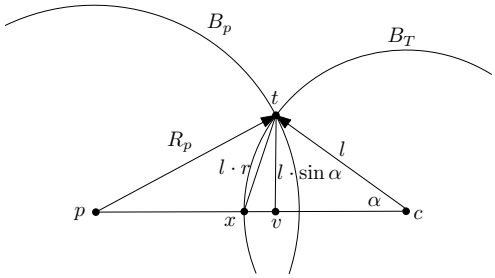
   (2a) the ball has the original radius R_p of B_p, or

   (2b) the ball touches the surface at another point. If this occurs we have found a point of M_{in} within distance Δ_1, and we are done.

3. In case (2a), we "roll" the new ball B'_p (with radius R_p) on the surface. More precisely, let K_1 be the component of $B_p \cap \mathcal{F}$ which contains x_0. Consider the balls of radius R_p that are tangent to $\mathcal{F}$ in a point of K_1 and lie on the same side of $\mathcal{F}$ as p. The locus of the centers of these balls is the inner parallel surface $\bar{F}$ of K_1. We claim that the rolling ball touches another point of $\mathcal{F}$, and therefore $\bar{F}$ contains a point of M_{in}.



(a) After shrinking and expanding the ball B_p we roll the new ball B'_p on K_1 (e.g. the gray ball).



(b) Deepest penetration into B_p

Figure 4.2:

We prove this by contradiction. Let us suppose that the ball can roll on K_1 without ever touching a second point of $\mathcal{F}$. K_1 cuts B_p into two parts: B^+ containing p, and the rest B^-. By Lemma 4.7 below, B^+ is completely covered by the tangent balls of K_1. Since by assumption these balls never hit another point of $\mathcal{F}$, it follows that K_1 is the only component of $\mathcal{F} \cap B_p$. Let $s \in B_p$ be the sample point whose pole is p. This point must lie on K_1 and therefore we can roll the empty tangent ball of radius R_p to s. The radius R_M of the medial ball at s is therefore at least R_p. On the other hand, each point of the medial axis is contained in the Voronoi cell of the nearest sample point, therefore $\|p - s\| = R_p \geq R_M$. This implies $R_p = R_M$ and the tangent ball at s has its center on M_{in}, and we are done. We remark that this last case can actually never arise, since $R_p > R_M$ unless the medial axis branches and the ball touches $\mathcal{F}$ in several points.

We have established that $\bar{F}$ contains a point m_x of M_{in} which is the center of a medial ball with radius R_p touching K_1 in x. We know by Lemma 4.6a that the angle $\gamma = \angle m_x x p$ is at most $3r + O(r)$. Thus, $\|p - m_x\| \leq R_p \cdot (3r + O(r^2))$.

□

In the following, we will assume that p is an inner pole. (The situation is symmetric for outer poles.)

Lemma 4.4. *Let p be a pole with polar radius R_p. The surface $\mathcal{F}$ cannot get closer to p than*

$$R_p\Big(\sqrt{1-4(r^2-\tfrac{r^4}{4})}-r^2\Big) \geq R_p\left(1-3r^2-O(r^4)\right).$$

For an r-sample with $r = 0.08$ the distance between the center p of a polar ball with radius R_p and $\mathcal{F}$ is larger than $0.9807 \cdot R_p$.

Proof. Let x be the point on $\mathcal{F}$ closest to p. Let B_T be an empty outer ball tangent to x with center c and radius $l = \mathrm{lfs}(x)$. By the sampling condition, there must be a sample t within distance rl of x. t lies outside the balls B_p and B_T and therefore the distance from x to the circle $\partial B_p \cap \partial B_T$ is at most $r \cdot l$ (see Figure 4.2(b)). Thus, the angle $\alpha = \angle cpt$ is bounded by $\sin\frac{\alpha}{2} \leq \frac{r}{2}$. For fixed l and R_p, the point x is closest to p when α is maximized. We thus analyze the situation for $\sin\frac{\alpha}{2} = \frac{r}{2}$:

$$\sin\alpha = 2\sin\frac{\alpha}{2}\cos\frac{\alpha}{2} \leq 2\cdot\frac{r}{2}\sqrt{1-\tfrac{r^2}{4}} = \sqrt{r^2-\tfrac{r^4}{4}}$$

$$\|v-p\| = \sqrt{R_p^2-(l\cdot\sin\alpha)^2} = \sqrt{R_p^2-l^2\cdot(r^2-\tfrac{r^4}{4})}$$

$$\|v-x\| = \sqrt{(l\cdot r)^2-(l\cdot\sin\alpha)^2} = \sqrt{(l\cdot r)^2-l^2\cdot(r^2-\tfrac{r^4}{4})} = \frac{l\cdot r^2}{2}$$

$$\|x-p\| \geq \|v-p\| - \|v-x\| \sqrt{R_p^2-l^2\cdot(r^2-\tfrac{r^4}{4})} - \frac{l\cdot r^2}{2}$$

The inner polar ball B_p contains a point of M_{in} ([ACK01, Corollary 13]), therefore $l \leq 2R_p$. It follows that the distance between p and $\mathcal{F}$ is at least

$$\sqrt{R_p^2-4\cdot R_p^2\cdot(r^2-\tfrac{r^4}{4})} \;-\; R_p \;\cdot\; r^2 \qquad = \quad R_p \;\cdot\; \Big(\sqrt{1-4\cdot(r^2-\tfrac{r^4}{4})} \;-\; r^2\Big),$$

as claimed in the lemma. □

Lemma 4.5. *Let B_p denote a polar ball with center p. For $r < 0.25$, the normal at a surface point $x \in B_p$ is never perpendicular to the ray $\overrightarrow{px}$.*

Proof. By contradiction: We assume that $\mathrm{lfs}(x) = l$. The points m_1, m_2 are centers of two tangent balls in x of radius l, both balls are empty, one is located inside $\mathcal{F}$ and the other one is outside. The surface must cross the path m_1pm_2 at least once. Assume it crosses m_1p in some point y. Then the following holds (see Figure 4.4):

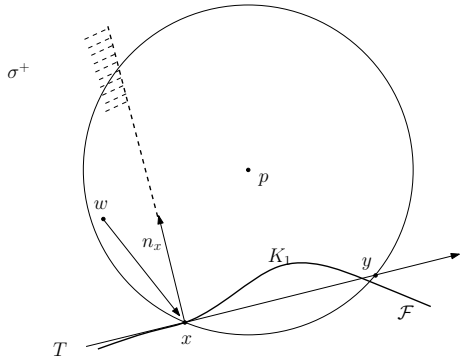


Figure 4.3: The tangent balls of K_1 cover B^+

$$\begin{aligned} R_p - \|p - m_1\| + \|m_1 - y\| &= R_p - \|y - p\| \\ &= \text{dist}(y, \text{boundary of } B_p) \\ &\leq \text{dist}(y, \text{closest sample}) \leq r \cdot \text{lfs}(y) \\ &\leq r(l + \|x - y\|) \leq r(l + \|x - m_1\| + \|m_1 - y\|) \\ &= r(2l + \|m_1 - y\|). \end{aligned}$$

Therefore,

$$\begin{aligned} \|p - m_1\| - R_p &\geq \|m_1 - y\|\,(1 - r) - 2rl \\ &\geq l(1 - r) - 2rl \geq l(1 - 3r) \end{aligned}$$

and also (according to Lemma 4.4)

$$\|p - m_1\| - l \geq \|p - y\| \geq R_p \left(\sqrt{1 - 4(r^2 - \tfrac{r^4}{4})} - r^2 \right).$$

If $\angle p x m_1$ is a right angle, then $\|p - m_1\|^2 = \|p - x\|^2 + l^2 \leq R_p^2 + l^2$. We have $l > 0, R_p > 0, \|p - m_1\| > 0$, so
$\|p - m_1\|^2 \geq (l(1-3r) + R_p)^2$ therefore we get $(l(1-3r) + R_p)^2 = R_p^2 + l^2$. For $l < 2R_p$ and $r \leq 0.25$ this leads to a contradiction. □

Lemma 4.6. *Let x be a surface point x inside a polar ball B_p with center p.*

a) The angle γ between $\overrightarrow{xp}$ and the surface normal at x is bounded by $3r + O(r^2) = O(r)$.

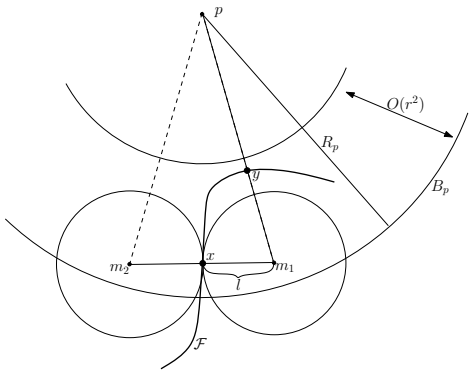


Figure 4.4: An absurd situation; the normal in x is perpendicular to the ray px. The points m_1 and m_2 can lie inside B_p (as in the picture) or outside.

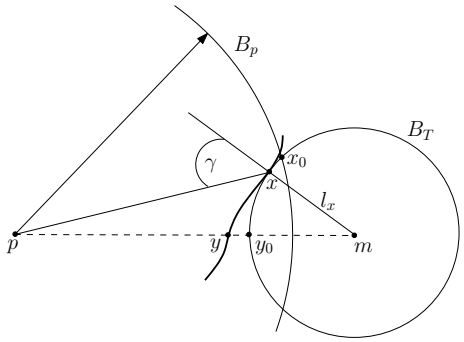


Figure 4.5: Bounding the angle γ. Intersection of the polar ball B_p and the empty tangent ball B_T.

b) *(The penetration bound) The distance from x to the boundary of B_p is is bounded by $\frac{3}{2}\,\mathrm{lfs}(x)(r^2 + O(r^3))$.*

Proof. Since the two parts have the same assumption, we start with calculations which are common to both claims. Consider the empty tangent ball B_T at x of radius $l_x = \mathrm{lfs}(x)$ on the opposite side of p and let m denote its center, see Figure 4.5. For any surface point z inside B_p the ray $\overrightarrow{pz}$ intersects the surface transversely and never tangentially by Lemma 4.5, therefore the surface patch around x must pass between p and B_T and cannot fold back. In particular, it must intersect the segment $\overline{mp}$ at some point y. The distance from y to the closest sample point s is $\|y - s\| \geq \|y - x_0\|$, where x_0 is the closest point of the circle $B_p \cap B_T$. By the sampling condition we know

$$\|y - x_0\| \leq \|y - s\| \leq r \cdot \mathrm{lfs}(y)$$

and using the Lipschitz condition we get

$$r \cdot \mathrm{lfs}(y) \leq r \cdot (\mathrm{lfs}(x) + \|x - y\|) \leq r \cdot (\mathrm{lfs}(x) + \|y - x_0\|)$$

Thus

$$\|y - x_0\| \leq \frac{r}{1-r} l_x$$

The intersection point $y = \overline{mp} \cap \mathcal{F}$ for which $\|y - x_0\|$ is smallest is y_0 with $\|y_0 - m\| = l_x$,

$$\|y_0 - x_0\| \leq \|y - x_0\| \leq \frac{r}{1-r} l_x.$$

We get for the angle $\alpha = \angle pmx_0$:

$$\alpha = \angle pmx_0 = 2 \arcsin \frac{\|y_0 - x_0\| / 2}{l_x} \leq 2 \arcsin \frac{r/2}{1-r}.$$

We now continue with the proof of part (a). A similar assertion [ACK01, Lemma 17] says that the angle at which B_T and B_p intersect is at most $2 \arcsin(2r)$, which is less than $\frac{\pi}{2}$ for $r < 0.35$.

Therefore, if x varies on the tangent sphere B_T, the largest possible γ (within B_p) is achieved when x is on x_0. From now on we assume $x = x_0$ (see Figure 4.6). We have

$$\gamma = \alpha + \beta. \qquad \sin \beta = \frac{l_x \sin \alpha}{R_p} \leq 2 \sin \alpha$$

because $l_x \leq 2R_p$ according to [ACK01, Corollary 13], so

$$\begin{aligned} \beta &\leq \arcsin(2 \sin \alpha) \\ \beta + \alpha &\leq 3r + 3r^2 + \tfrac{33}{8} r^3 + \tfrac{51}{8} r^4 + O(r^5) = 3r + O(r^2) \\ \gamma &\leq 3r + O(r^2) = O(r), \end{aligned}$$

as claimed in the lemma.

(b) The distance d from x to B_p is no more than the distance of y_0 to B_p. We can therefore use Figure 4.6:

$$\begin{aligned} d &= R_p(1 - \cos \beta) + l_x(1 - \cos \alpha) \\ &= R_p(1 - \sqrt{1 - \sin^2 \beta}) + l_x(1 - \sqrt{1 - \sin^2 \alpha}) \\ &= R_p(1 - \sqrt{1 - \sin^2 \alpha \cdot l_x^2 / R_p^2}) + l_x(1 - \sqrt{1 - \sin^2 \alpha}) \\ &= \tfrac{1}{2} \cdot [R_p(\sin^2 \alpha \cdot l_x^2 / R_p^2 + O(\alpha^4)) + l_x(\sin^2 \alpha + O(\alpha^4))] \\ &= \tfrac{1}{2} \cdot [(\sin^2 \alpha \cdot l_x(l_x / R_p) + O(\alpha^4)) + l_x(\sin^2 \alpha + O(\alpha^4))] \\ &\leq \tfrac{1}{2} \cdot [(\sin^2 \alpha \cdot 2l_x + O(\alpha^4)) + l_x(\sin^2 \alpha + O(\alpha^4))] \\ &= l_x \cdot (\tfrac{3}{2} r^2 + 3r^3 + O(r^4)). \end{aligned}$$

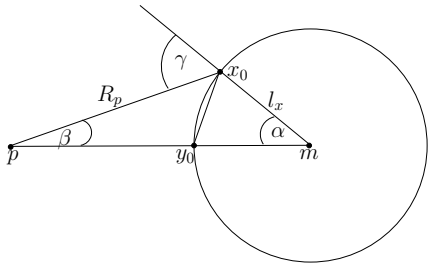


Figure 4.6: Bounding the angle γ.

□

To complete the proof of Theorem 4.3, we still need to show that the tangent balls of K_1 cover all parts of B^+. Recall that K_1 cuts B_p in two parts: B^+ containing p, and the rest B^-.

Lemma 4.7. *The tangent balls of K_1 completely cover B^+.*

Proof. Let $w \in B^+$ and let x be the closest point of K_1. We claim that the tangent ball at x covers w. If x lies in the interior of K_1, then wx is perpendicular to $\mathcal{F}$, and the claim is obvious. Let us assume that x is at the boundary of K_1, that is $B_p \cap \mathcal{F}$ (see Figure 4.3). Assume that the surface normal n_x does not go through p; otherwise it is obvious that w is covered. Consider the plane σ through n_x and through the point p. Figure 4.3 shows the projection on this plane. Locally around x, $\mathcal{F}$ is approximated by the tangent plane T and $B_p \cap \mathcal{F}$ is the halfspace of T that projects onto the ray xy in Figure 4.3. It follows that x can only be the point of K_1 closest to w, if w lies in the plane σ and in the closed halfplane σ^+ of σ which is bounded by n_x and does not contain p. □

4.7 Construction of balls

4.7.1 Surface balls

In order to maintain correct topology of the piecewise linear surface reconstruction, the surface balls we generate have to be large enough such that their union does not only cover S but also $\mathcal{F}$ and, on the other hand, these balls avoid the medial axis of the union of the balls in $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$ and $\mathrm{DMT}(\mathcal{O})_{\mathrm{out}}$. The above restrictions limit the possible radii to a certain range. Maximizing the radii within this range will lead to a coarse result (which is desirable for seed polytopes), while minimizing the radii of

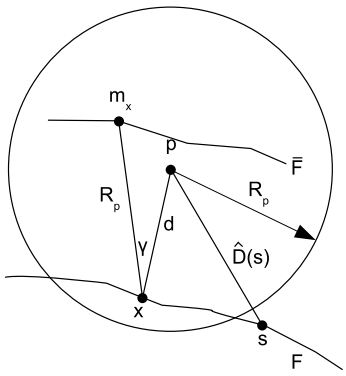


Figure 4.7: Distance from pole p to the medial axis point m_x

the surface balls will lead to a faithful and detailed representation of the object. The choice of the radii determines the degree by which the surface balls are pruned in a subsequent set covering step.

Lower bound on the radii

To ensure that $\mathcal{F}$ is completely covered by surface balls we choose the radii of the surface balls such that they cover at least the intersection of their site's Voronoi cells with $\mathcal{F}$. For a point s in an r-sample, this intersection is covered by a sphere around s whose radius is $\rho \geq \frac{r}{1-r} \cdot \text{lfs}(s)$, see [AB99], and so the surface balls need to have at least that radius. As $\text{lfs}(s)$ is unknown, we need to estimate it in terms of the distance $\hat{D}(s)$ between s and the nearest among the poles of all sample points. Using Lemma 4.8 below, we get

$$\text{lfs}(s) \leq 1.2802 \cdot \hat{D}(s)$$

and so we must choose the radius ρ of a surface ball around s to be at least

$$\rho \geq \tfrac{r}{1-r} \cdot 1.2802 \cdot \hat{D}(s).$$

The distance $\hat{D}(s)$ can be calculated relatively easily using a spatial search structure.

Lemma 4.8. *Let $s \in S$ be a point of an r-sample S with $r \leq 0.08$, and let $\hat{D}(s) = \|s - p\|$ denote its distance to the nearest pole p. Then*

$$\text{lfs}(s) \leq 1.2802 \cdot \hat{D}(s).$$

Proof. The local feature size of s cannot be larger than $\hat{D}(s)$ plus the distance from p to the medial axis. To bound the latter distance for a specific value of r, we revisit the

cases developed in Theorem 4.3 (and we use the notation introduced there). If case (2a) occurs we know that $\bar{F}$ contains a point $m_x \in \mathrm{M_{in}}$ ($\mathrm{M_{out}}$); see Figure 4.7. By Lemma 4.6a, the maximum angle between the touching point $x \in K_1$ of the medial ball centered at m_x and p is $\gamma = \angle m_x x p < 14.99°$ if $r \leq 0.08$. By Lemma 4.4,

$$d = \|x - p\| \geq \left(\sqrt{1 - 4(r^2 - \tfrac{r^4}{4})} - r^2 \right) \cdot R_p > 0.9807 \cdot R_p.$$

Therefore

$$\|p - m_x\| \leq 2 \cdot R_p \cdot \sin(\frac{\gamma}{2}) + (1 - 0.9807) R_p < 0.2802 \cdot R_p$$

which is at most $0.2802 \cdot \hat{D}(s)$ because s lies outside the polar ball centered at p. Otherwise, case (2b) occurs and by Lemma 4.4, p is not farther from $\mathrm{M_{in}}$ ($\mathrm{M_{out}}$) than

$$R_p \cdot (1 - \sqrt{1 - 4 \cdot (r^2 - \tfrac{r^4}{4})} + r^2) < 0.0193 \cdot R_p.$$

The lemma follows. □

Upper bound on the radii

To prevent surface balls from "different" parts of $\mathcal{F}$ from intersecting we want to ensure that they do not reach the discrete medial axis $\mathrm{DM_{in}}$ (resp. $\mathrm{DM_{out}}$). Thus, the discrete local feature size $\widetilde{\mathrm{lfs}}(s)$ is an upper bound on the radius that we can use. We will replace $\widetilde{\mathrm{lfs}}(s)$ by a smaller value, that is easier to compute, see Proposition 4.2.

Consequently, the minimum distance from s to any of the two weighted zero α-shapes is a lower bound on $\widetilde{\mathrm{lfs}}(s)$. Computing $\mathcal{A}_{\mathrm{in}}$ and $\mathcal{A}_{\mathrm{out}}$ and determining the minimum distance directly would consume too much time and memory, however. We show how to estimate this distance, again using the distance $\hat{D}(s)$ to the nearest pole to s.

Lemma 4.9. *Let s be a sample point, and let v be a point with the following properties*

- *v lies in the Voronoi cell of s.*
- *v is not in the interior of the polar ball around the pole p of s that lies on the same side of $\mathcal{F}$ as v.*

Then

(a) $\|v - s\| = O(r) \cdot \mathrm{lfs}(s)$. *In particular, for $r = 0.08$, the distance to s is at most* $0.123 \cdot \mathrm{lfs}(s)$.

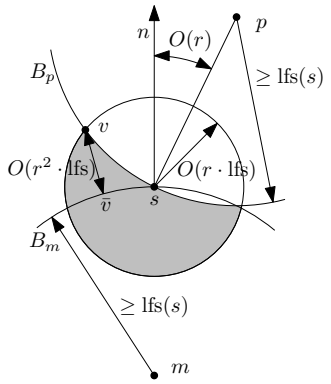


Figure 4.8: A point v that is not covered by the polar ball must lie close to the surface.

(b) *The distance from v to the closest point $\bar{v}$ on the surface is*

$$O(r^2 \operatorname{lfs}(s)) = O(r^2 \operatorname{lfs}(\bar{v})).$$

For $r = 0.08$, the distance $\|v - \bar{v}\|$ is at most $0.0355 \cdot \operatorname{lfs}(s) \leq 0.0424 \cdot \operatorname{lfs}(\bar{v})$.

Proof. We perform the calculation for $r = 0.08$, and only indicate the asymptotic dependence on r. We will first show part (a).

$$\|v - s\| \leq 0.123 \cdot \operatorname{lfs}(s) = O(r \operatorname{lfs}(s)).$$

Let p be the pole of s on the same side of the surface as v. If $\|v - s\| > kr \cdot \operatorname{lfs}(s)$ for $k = 1.536$, the angle between sv and the surface normal is at most

$$\arcsin \frac{1}{k(1-r)} + \arcsin \frac{r}{1-r} < 47.2^\circ$$

see [AB99, Lemma 4]. Similarly, the angle between the normal and sp is at most $2 \arcsin \frac{r}{1-r} < 12.8^\circ$. In total the angle vsp is less than 60°. Since $\|v - s\| \leq \|p - s\|$, by the definition of the pole, it follows that v must be contained in the polar ball around p, whose radius is $\|p - s\|$, a contradiction. We thus conclude that v is contained in a ball of radius

$$kr \cdot \operatorname{lfs}(s) \leq 0.123 \cdot \operatorname{lfs}(s) \quad (= O(r \operatorname{lfs}(s)))$$

around s. Since v avoids the polar ball B_p around p, it lies in the shaded region indicated in Figure 4.8. The direction sp of the polar ball deviates at most $2 \arcsin \frac{r}{1-r} < 12.8^\circ$ $(= O(r))$ from the normal direction n at s. Thus the "highest" possible position of v is as indicated in the figure. We know that the surface must pass above the opposite medial

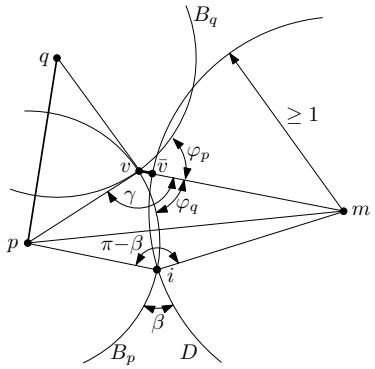


Figure 4.9: Schematic figure of an intersection of two polar balls such that their intersection point v is not covered by the union of polar balls.

ball P_m of s, and thus we can estimate the distance from v to the surface and prove (b). A straightforward calculation gives the bound $\|v - \bar{v}\| \leq 0.0355 \operatorname{lfs}(s)$ $(= O(r^2 \operatorname{lfs}(s)))$. By the Lipschitz condition,

$$0.0355 \operatorname{lfs}(s) \leq \frac{0.0355}{1 - 0.123 - 0.0355} \operatorname{lfs}(\bar{v}) \leq 0.0424 \cdot \operatorname{lfs}(\bar{v})$$

is obtained. □

Lemma 4.10. *Let pq be an edge of the weighted zero α-shape $\mathcal{A}_{\text{in}}$ ($\mathcal{A}_{\text{out}}$). Then the exterior angle of intersection between the polar balls B_q, B_p around p and q is at least* $120°$.

Proof. Since pq is an edge of the weighted zero α-shape, there is a point v on the intersection of the boundaries of the two polar balls B_p and B_q which is not covered by any other polar ball, see Figure 4.9. Therefore, the neighborhood of v contains points outside all polar balls and, by Lemma 4.9 v is close to $\mathcal{F}$: For the closest surface point $\bar{v}$ we have

$$d = \|v - \bar{v}\| \leq 0.0424 \cdot \operatorname{lfs}(\bar{v}).$$

Without loss of generality, we assume $\operatorname{lfs}(\bar{v}) = 1$. Consider the medial ball B of $\bar{v}$ on the opposite site, with center m and radius $\|\bar{v} - m\| \leq \operatorname{lfs}(\bar{v}) = 1$. By [ACK01, Lemma 17], a polar ball B_p or B_q intersects a medial ball D on the opposite site at angle $\beta \leq 2 \arcsin 2r$. Let us focus on one ball B_p and the angle ϕ_p between this ball and the surface normal vm. The other ball is treated in the same way, and the total exterior angle is then $\phi_p + \phi_q$.

We have $\phi_p = \gamma - \pi$, where $\gamma = \angle pvm$. To get an upper bound on ϕ_p (or on γ), let us fix the angle γ and try to find circles B_p and D that are consistent with this situation. We have the following constraints:

(i) $1 = \mathrm{lfs}(\bar{v}) \geq \|\bar{v} - m\|$;

(ii) $d := \|v - \bar{v}\| \leq 0.0424 \cdot \mathrm{lfs}(\bar{v}) \leq 0.0424$;

(iii) The intersection angle between B_p and D is $\beta \leq 2 \arcsin 2r$.

This gives us a distance $\|c - v\| = 1 + d$, using the triangle inequality we get $\|q - v\| = 1 - d$. For the triangle qcv only the segment qc is of unknown length. We consider also a second triangle, formed by the points q, c and one intersection point i of the medial ball with the polar ball B_q. Again only the distance of the segment qc is unknown. From the triangles we get the following equations:

$$\cos\beta = \frac{1 + (1-d)^2 - \|c-q\|^2}{2(1-d)},$$
$$\cos\gamma = \frac{(1+d)^2 + (1-d)^2 - \|c-v\|^2}{2(1-d)(1+d)},$$

for $\beta = \angle cvq = \pi - \beta = \pi - 2 \arcsin 2r$, $\gamma = \angle qic$, $d = 0.0355$. Solving these equations for γ gives an angle $\varphi = 2 \cdot (\gamma - \pi/2) > 120°$. □

Based on the preceding lemmas, it is possible to derive the following bound on $\widetilde{\mathrm{lfs}}(s)$.

Lemma 4.11. *If m is a point on an edge pq of $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$ (or in a triangle pqr of $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$) and v is outside or on the boundary of $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}})$ then*

$$\|m - v\| \geq 0.817 \cdot \min\{\|p - v\|, \|q - v\|\},$$

(or $\|m - v\| \geq 0.817 \cdot \min\{\|p - v\|, \|q - v\|, \|r - v\|\}$, respectively).

Proof. We first consider the case when m lies on an *edge pq*,as illustrated in Figure 4.10. Let m' be the point on pq that is closest to v. If m' is one of the endpoints p or q, we are done:

$$\|m - v\| \geq \|m' - v\| = \min\{\|p - v\|, \|q - v\|\}.$$

Otherwise we know that $m' - v$ is perpendicular to pq. We know from Lemma 4.10 that the intersection of the two polar balls B_p and B_q cannot be too thin: their angle of intersection is at least 120°. For fixed balls B_p and B_q, the angles and hence the

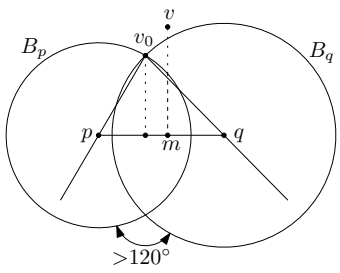


Figure 4.10: The distance from the sample point s to the weighted zero α-shape

ratios are minimized when s lies on the intersection between the balls (the point v_0 in the figure).

Now keeping v_0 fixed at the intersection and considering a variation of the balls B_p and B_q, maintaining $\min\{\|v-p\|, \|v-q\|\}$, it is clear that the distance from v to the edge pq is minimized when the angle $\angle pvq$ is at its upper bound of $60°$ and the two distances are equal: $\|v-p\| = \|v-q\|$. Then the ratio $\|v-v\|/\|v-p\| = \cos 30° > 0.866$.

Now consider the case when m lies in a *triangle* pqr. If the point m' on pqr that is closest to v lies on an edge or at a vertex of the triangle, we have reduced the problem to the previous case. Otherwise we know that $m' - v$ is perpendicular to pqr. The remaining argument is similar as in the case of an edge: The extreme situation is a triangular pyramid with equal angles $\angle pvq = \angle qvr = \angle rvp = 60°$ at the apex m and equal sides $\|p-v\| = \|q-v\| = \|r-v\|$. The ratio between the height of this pyramid and the length $\|p-v\|$ is $\sqrt{(1+2\cos 60°)/3} > 0.817$. □

Corollary 4.12. *Let $s \in S$ be a sample point, and let $\hat{D}(s)$ be its distance to the nearest pole. Then*

$$\hat{D}(s) \geq \widetilde{\mathrm{lfs}}(s) \geq 0.817 \cdot \hat{D}(s).$$

Proof. Since the poles are part of the discrete medial axis, the inequality $\widetilde{\mathrm{lfs}}(s) \leq D(s)$ is obvious. For the other direction, we bound $\widetilde{\mathrm{lfs}}$ by the distance from v to the weighted zero α-shape $\mathcal{A}$ of the polar balls, which contains the discrete medial axis. The proof of the lower bound on the ratio

$$\frac{\widetilde{\mathrm{lfs}}(v)}{D} = \frac{\|v-m\|}{D} \geq \max\left\{\frac{\|v-m\|}{\|v-p\|}, \frac{\|v-m\|}{\|v-q\|}\right\},$$

follows from Lemma 4.11. □

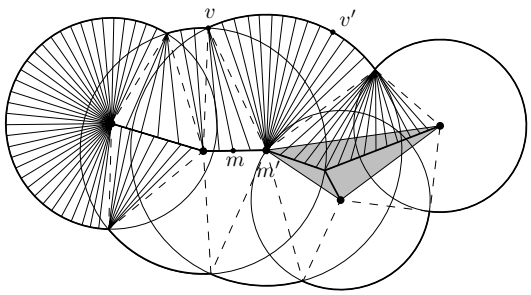


Figure 4.11: Part of the fibration which is used to show isotopy. The shaded area is the weighted zero α-shape.

4.7.2 Topological correctness

To show that the union $U(B_{\mathcal{F}})$ of surface balls is homotopy-equivalent to the surface $\mathcal{F}$, we follow the standard approach of using a fibration (a partition of $U(B_{\mathcal{F}})$ into a continuous family of curves, each intersecting $\mathcal{F}$ in a single point) and moving the boundaries of $U(B_{\mathcal{F}})$ along the fibers towards $\mathcal{F}$.

The usual fibration by surface normals does not work since the medial axis might be closer than it appears from looking at the sample points, see Figure 4.1. Instead we use the fibers of the union $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}})$ of all polar balls. It is known that this union is homotopy-equivalent to $\mathcal{O}$, and its boundary is homotopy-equivalent to $\mathcal{F}$ [AK00].

The boundary of the union $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}})$ is not smooth, but still, it is in a certain sense "smooth from the inside" (it has no convex edges or vertices) and has therefore a reasonable fibration connecting the boundary to its inner medial axis $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$, see Figure 4.11. We concentrate on the inner discrete medial axis $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$; the outer discrete medial axis $\mathrm{DMT}(\mathcal{O})_{\mathrm{out}}$ is treated analogously. The fibers are line segments that partition $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}) \setminus \mathrm{DM}_{\mathrm{in}}$, and they run from a surface point v on the boundary to a point m on the inner discrete medial axis $\mathrm{DM}_{\mathrm{in}}$. In three dimensions, there are three types of fibers: from a point v on a spherical patch of the boundary to a vertex m of the medial axis; from a point v on a circular edge formed as the intersection of two spheres to a point m on an edge of the medial axis; and from a vertex v of the boundary, formed as the intersection of three (or more) spheres to a point m on a face of the medial axis. Our proof treats all three cases uniformly.

We take the radius of the surface balls as $\rho\hat{D}(s)$ where the factor ρ can be chosen in the interval

$$\rho_{\min} = 0.24 \leq \rho \leq \rho_{\max} = 0.56. \tag{4.1}$$

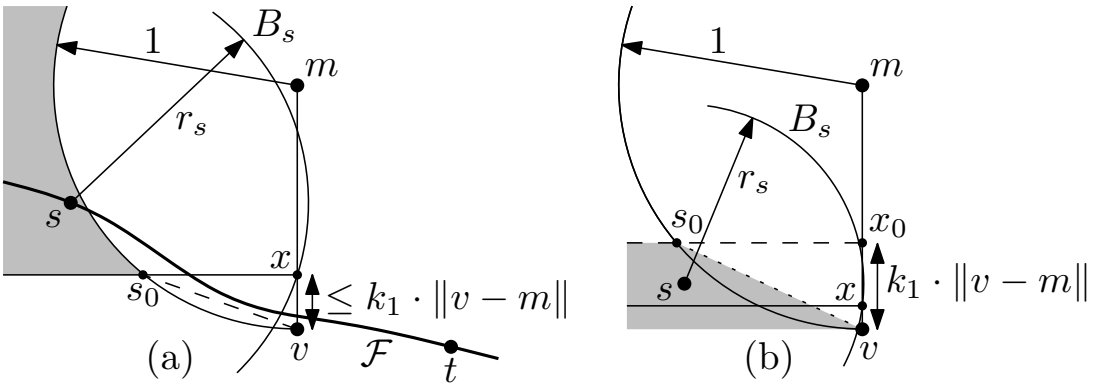


Figure 4.12: A ball B_s that intersects the fiber vm improperly

The upper bound ensures that the surface balls do not intersect the discrete medial axis, and the lower bound ensures that they are large enough to cover the surface completely. The bounds are stricter that would be required to reach only these two goals, since we also want to achieve ensure topological correctness of the union $U(B_\mathcal{F})$ of surface balls:

Lemma 4.13. *If ρ is chosen in the interval* (4.1)*, every fiber from a point v on the boundary of $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}})$ to a point m on the medial axis of $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}})$ starts in the union $U(B_\mathcal{F})$ of surface balls and intersects the boundary of $U(B_\mathcal{F})$ precisely once.*

The lemma implies that the boundary of $U(B_\mathcal{F})$ can be continuously deformed along the fibers into the boundary of $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}})$, and thus the two boundaries are homotopy-equivalent. The boundary of $U(\mathrm{DMT}(\mathcal{O})_{\mathrm{in}})$ is already known to be homotopy-equivalent to $\mathcal{F}$, and thus, the correct topology is established.

Proof. For simplicity we prove the bound for $\rho = 0.3$. The calculation for general ρ is slightly more involved.

Let B_s be a surface ball around a sample point s such that the segment vm enters B_s in a point x, see Figure 4.12a. We will show that this does not lead to a violation of the lemma, because the segment vx is covered by the union of surface balls. We assume without loss of generality that vm is vertical and $\|m - v\| = 1$. We first show that x must have distance $\|x - v\| \leq k_1$ for $k_1 = 0.074$.

Suppose that this is not true. The medial ball of radius 1 around m is inside the union of balls, and hence it does not contain s: $\|s - m\| \geq 1$. We claim that this implies

$$\|s - x\| > 0.37 \cdot \|s - m\|. \tag{4.2}$$

We know that s must lie outside the ball of radius 1 around m; s must also lie above the horizontal line through x. Thus, s is restricted to the shaded area in the figure. The ratio $\|s-x\|/\|s-m\|$ is minimized when x is as low as possible ($\|x-v\| = k_1$) and s is at the lower right corner s_0 of this area. Here we have $\|s-x\|^2 + (1-k_1)^2 = 1$, from which one can compute $\|s-x\|/\|s-m\| = \|s-x\| > 0.37$.

On the other hand, since $m \in \mathrm{DMT}(\mathcal{O})_{\mathrm{in}} \subseteq \mathcal{A}_{\mathrm{in}}$, we have by definition $\|s-m\| \geq \widetilde{\mathrm{lfs}}(s) \geq 0.817\hat{D}(s)$, by Lemma 4.11. Thus, the radius r_s of B_s is $r_s = \|s-x\| \leq \rho\hat{D}(s) \leq \rho/0.817 \cdot \|s-m\| < 0.368 \cdot \|s-m\|$, contradicting (4.2).

Let us denote the extreme positions of s and x in the above analysis by s_0 and x_0.We have established that x and s lie below horizontal line s_0x_0, see see Figure 4.12b. For an arbitrary x and s we now claim

$$\frac{\|s-x\|}{\|x-v\|} \geq \frac{\|s_0-x_0\|}{\|x_0-v\|} \geq 5. \tag{4.3}$$

We know that s must always lie higher than x, For a fixed point x, we can rotate s around x until it lies at the same height as x, without changing the above ratio, So we can assume that s and x lie at the same height, with $\|x-v\| \leq k_1$. The sample s cannot lie in the polar ball around m, and in particular, s must lie below the dotted line segment. The claim (4.3) follows.

Now to complete the proof we will show that the segment vx is covered by a surface ball, namely by the ball around the surface sample t closest to v. We are done if we can show that the radius r_t of this ball is at least $\|t-v\| + \|v-x\|$:

$$r_t = \rho\hat{D}(t) \geq \|t-v\| + \|v-x\| \tag{4.4}$$

This implies that $r_t \geq \|t-v\|$ and $r_t \geq \|t-x\|$ (by the triangle inequality), and thus ensures that the whole segment vx is covered. It establishes also that the starting point v of the fiber is covered, irrespective of whether another ball B_s intersects vm "in an improper way".

First we show that there is a sample point t with

$$\|t-v\| \leq 0.123 \cdot \mathrm{lfs}(t) \tag{4.5}$$

We distinguish two cases:

(a) v lies inside $\mathcal{F}$ (on the same side as m), see Figure 4.13(a). Let t be the sample point closest to v. The point v satisfies the assumptions of Lemma 4.9 with respect to t: By definition, v lies in the Voronoi cell of t. Moreover, v lies in none of the polar balls around the vertices of $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$. Thus, by Lemma 4.9a, $\|t-v\| \leq 0.123 \cdot \mathrm{lfs}(t)$.

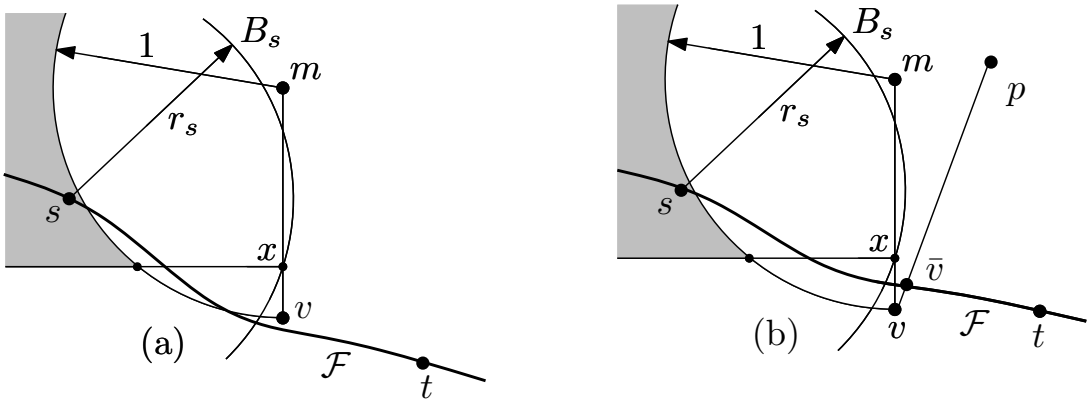


Figure 4.13: A ball B_s that intersects the fiber vm improperly, v lies either inside $\mathcal{F}$ (a) or outside $\mathcal{F}$ (b)

(b) v lies outside $\mathcal{F}$, see Figure 4.13(b). By Lemma 4.11, there is a pole p in $\mathrm{DMT}(\mathcal{O})_{\mathrm{in}}$ such that

$$\|p - v\| \leq \frac{1}{0.817} \cdot \|m - v\| \leq 1.224 \cdot \|m - v\|$$

The segment vp must intersect $\mathcal{F}$ in some point $\bar{v}$. Lemma 4.6b limits the penetration of the surface point $\bar{v}$ into the ball B_p:

$$\|\bar{v} - v\| \leq (3/2 \cdot r^2 + O(r^3)) \cdot \mathrm{lfs}(\bar{v}).$$

In particular, for $r = 0.08$,

$$\|\bar{v} - v\| \leq 0.0114 \cdot \mathrm{lfs}(\bar{v}).$$

The nearest sample point t from $\bar{v}$ is less than $r \cdot \mathrm{lfs}(t)$ away:

$$\|\bar{v} - t\| \leq r \cdot \mathrm{lfs}(t)$$

The Lipschitz condition yields

$$\mathrm{lfs}(\bar{v}) \leq \mathrm{lfs}(t) + \|\bar{v} - t\| \leq (1 + r) \cdot \mathrm{lfs}(t).$$

Therefore we get:

$$\begin{aligned}
\|t - v\| &\leq \|v - \bar{v}\| + \|\bar{v} - t\| \\
&\leq 0.0114 \cdot \mathrm{lfs}(\bar{v}) + r \cdot \mathrm{lfs}(t) \\
&\leq 0.0114 \cdot (1 + r)\,\mathrm{lfs}(t) + r \cdot \mathrm{lfs}(t) \\
&\leq 0.093\,\mathrm{lfs}(t) \leq 0.123\,\mathrm{lfs}(t)
\end{aligned}$$

proving (4.5).

We have, by Lipschitz continuity, and using (4.3),

$$\begin{aligned}\hat{D}(t) &\geq \hat{D}(s) - \|s-x\| - \|x-v\| - \|v-t\| \\ &\geq \|s-x\|/\rho - \|s-x\| - \|x-v\| - \|v-t\| \\ &= \|s-x\|(1/\rho - 1) - \|x-v\| - \|v-t\| \\ &\geq 5(1/\rho - 1)\|x-v\| - \|x-v\| - \|v-t\| \\ &= [5(1/\rho - 1) - 1] \cdot \|x-v\| - \|v-t\| \\ &> 10.6 \cdot \|x-v\| - \|v-t\| \end{aligned} \tag{4.6}$$

By (4.5) and Lemma 4.8, we have $\|v-t\| \leq 0.123 \cdot \mathrm{lfs}(t) \leq 0.123 \cdot 1.2802 \cdot \hat{D}(t) < 0.1575\hat{D}(t)$ and hence

$$\hat{D}(t) > 6.3 \cdot \|v-t\| \tag{4.7}$$

Multiplying (4.6) by 0.095, (4.7) by 0.175, and adding them together yields

$$0.27\hat{D}(t) \geq \|x-v\| + \|v-t\|, \tag{4.8}$$

implying (4.4). □

4.8 Pruning by set covering

If we have a sample that is much denser than required by our conditions, we will get a correct "surface reconstruction", but we would like to obtain a coarser approximation to reduce the data, while maintaining topological correctness. We will therefore only use a subset of the surface balls.

We establish a condition that is easy to check and guarantees the correct topology: As before, we use balls of radius $\rho\hat{D}(u)$ around surface points u; for each ball we also consider a shrunk copy of radius $\bar{\rho}\hat{D}(u)$, where $\bar{\rho} = 0.03 < \rho$. We can then prove the following statement.

Theorem 4.14. *If the shrunk balls around the points u of a subset $S' \subseteq S$ cover all sample points S, then the union of the original balls (of radius $\rho\hat{D}(u)$) around these points is homotopy-equivalent to $\mathcal{F}$.*

Proof. The proof proceeds via the statement of Lemma 4.13. In that proof, we have established the existence of a sample point t that is close enough to v such that the ball around t covers the segment vx. This is extended to the present setting as follows: we

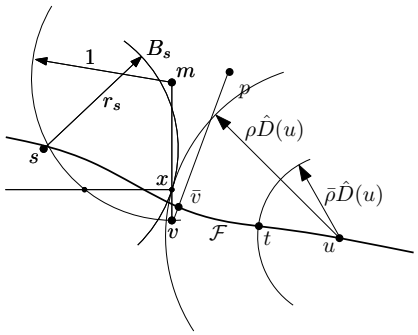


Figure 4.14: The segment vm is covered by the enlarged ball around u.

can now no longer be sure that the ball around t is used, but there must be a (shrunk) ball around some sample point u that covers t. Then the (original) ball around u is large enough to guarantee that it reaches vx.

We know, by the pruning condition, that the covering contains a ball of radius $\rho\hat{D}(u)$ around a sample point u such that the shrunk ball with radius $\bar{\rho}\hat{D}(u)$ covers t:

$$\|u - t\| \leq \bar{\rho}\hat{D}(u)$$

From this, together with the above bound (4.8) on $\|t - x\|$, we obtain

$$\|u - x\| \leq \|u - t\| + \|t - x\| \leq \bar{\rho}\hat{D}(u) + (\rho - \bar{\rho})\hat{D}(u) = \rho\hat{D}(u),$$

and thus the ball B_u covers x. The same calculation shows that B_u covers v, and hence the whole segment vx. □

We try to select a minimum subset of surface balls whose shrunk copies cover the whole sample. This is an instance of the (in general NP-hard) set covering problem, and in Section 3.6.1 we describe a combination of exact and heuristic methods that yields high quality results.

The lemma remains true if the shrinking factor 0.03 is replaced by a smaller number. This parameter allows us to scale the algorithm to different levels of coarseness or refinement of the approximation. If the shrinking factor approaches 0, each shrunk ball will contain no sample points except its center, and thus the full sample will be used. The small radius $0.03 \cdot \hat{D}$ that we have proven may not seem very impressive, but it must be seen in relation with the sampling constant $r = 0.08$. Thus, balls will start to be eliminated as soon as the actual sampling density exceeds the required minimum by a factor of about 4–5 (in terms of the sampling radius). Note, that unlike to what we are able to prove until now, shrinking factors of up to 0.6 work well in practice.

4.9 Experimental results

4.9.1 Example 1

Figure 4.15 illustrates how different choices of radii for surface balls lead to different levels of detail in the approximating polyhedral surface mesh. The initial point cloud for this *doubleTorus* model consists of 85237 points. Due to the effect of pruning, the mesh for the big ring is more and more coarsened, whereas the necessary details are preserved for the small ring. The running times for these computations (for a single threaded application on a Core2 Duo E6700 CPU) are shown in Table 4.1. Filtered floating point arithmetic has been used.

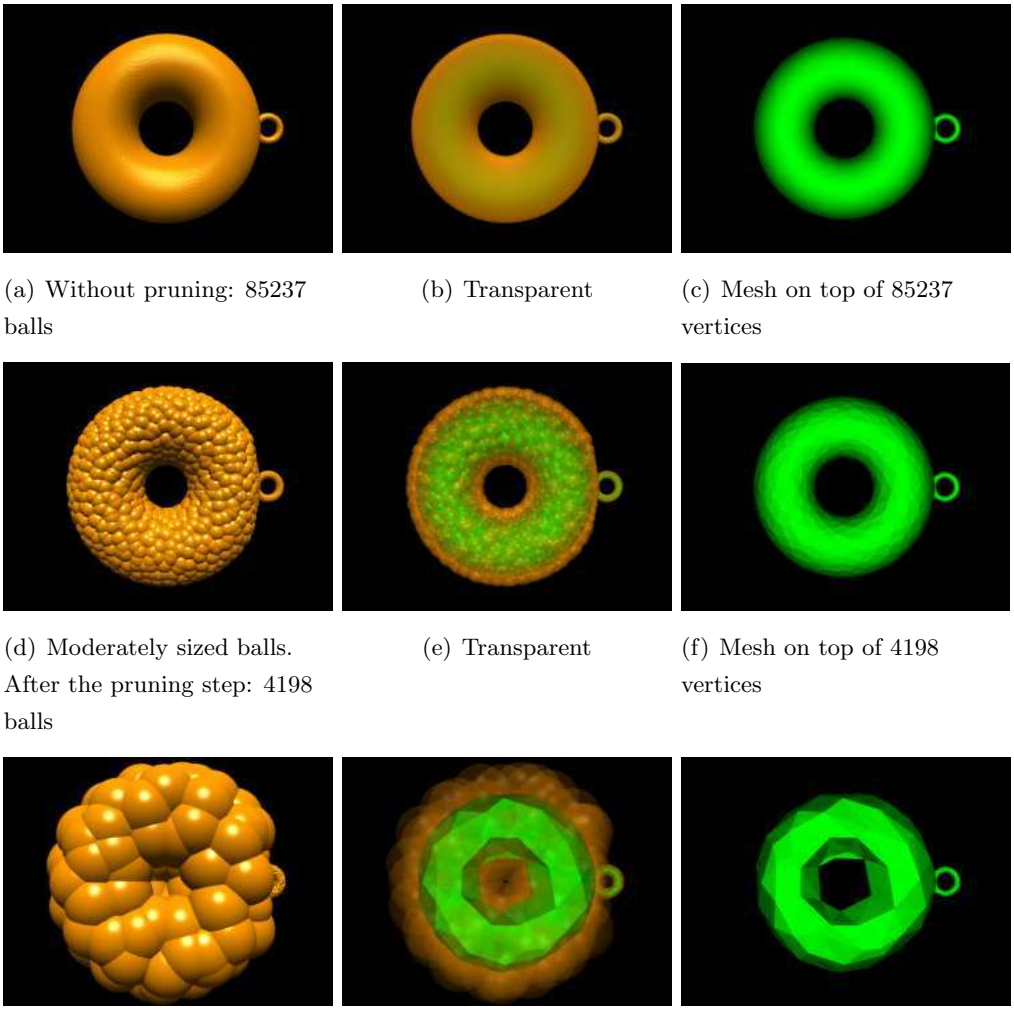
Figure	4.15abc	4.15def	4.15ghi
Surface balls	55s	55s	55s
Pruning	-	35s	159s
# Remaining balls	85237	4198	549
Weighted zero α-shape	217s	7s	1s

Table 4.1: Runtimes for the *doubleTorus* model in Figure 4.15

4.9.2 Real World Datasets

The input points for the *doubleTorus* model in Section 4.9.1 where taken from a smooth model, such that the r-sampling condition was kept. This condition is important for our theoretic work, because our proofs build on existing work, [ACK01, AK00], that also uses r-samplings. Besides, in lack of any restriction on sampling quality one could as well call a set of, say, four sample points a proper sample of the *doubleTorus* model, so the r-sampling condition makes definitely sense.

As to the situation in real world settings: When an object is scanned, the data points are usually acquired in a sequence of overlapping scans, between which the scanner or the object is moved for full coverage of the object. The independent scans, each afflicted with noise and inaccuracies, must then be registered, aligned and merged. The merged point cloud, which is the input for the surface reconstruction step, will most likely contain errors from misalignment, noise, and inaccuracies as well as holes, i.e. missing data, due to an inappropriate scanning angle. Thus, we can not expect r-sampled inputs in a real world setting.



(a) Without pruning: 85237 balls

(b) Transparent

(c) Mesh on top of 85237 vertices

(d) Moderately sized balls. After the pruning step: 4198 balls

(e) Transparent

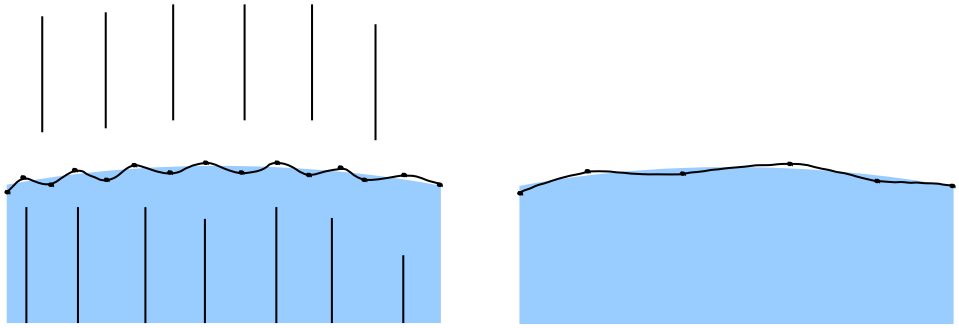
(f) Mesh on top of 4198 vertices

(g) Large balls allow for heavy pruning: 549 balls

(h) Transparent

(i) Mesh on top of 549 vertices

Figure 4.15: Reconstruction of the *doubleTorus* model



(a) Noisy point sample, high density (b) Noisy point sample, low density

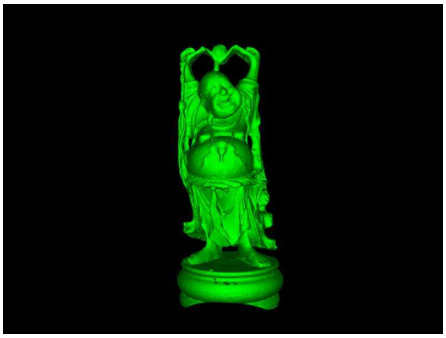
Figure 4.16: Effect of sampling density and noise

In Section 4.9.3 we will reconstruct the well known *Happy Buddha* model. The scan data for this model is available from the Stanford 3D Scanning Repository, [STA], where it is provided in four different resolutions, namely 543652, 144647, 32328 and 7108 vertices. One may assume that the point cloud with the highest density is best suited for surface reconstruction. But ironically, in presence of noise and other errors, dense point samples can be worse than sparse ones. In Figure 4.16(a) we sketch a dense, noisy point sample of an object. This point sample appears to the Voronoi approach as if it comes from a feature-rich surface. The medial axis of such an object would comprise branches running to these small features. The corresponding (discrete) medial axis transform of the object must express these by small, surface-near balls. In point samples with lower density this effect fades away, as sketched in Figure 4.16(b).

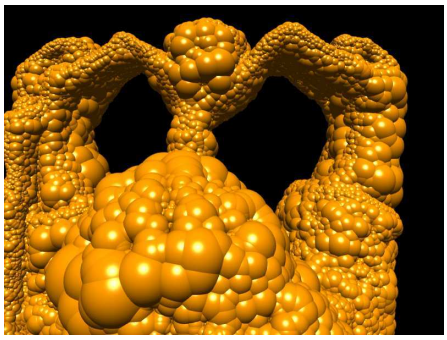
4.9.3 Example 2

According to Equation 4.1, the radii of the balls in $B_{\mathcal{F}}$ can be chosen between $0.24 \cdot \hat{D}(s_i)$ and $0.56 \cdot \hat{D}(s_i)$. This range guarantees topological correctness as long as the input quality is sufficient. However, for non-r-sampled inputs, there is no guarantee. Thus, we have made our application such that the parameters from our theoretic work can be overruled, i.e., adapted to the given data. Figure 4.17 shows the pruned set of surface balls $B_{\mathcal{F}}$ and a reconstructed mesh of the *Happy Buddha* dataset, which consists of 543652 vertices. In this example, the effects described in Section 4.9.2 led to very small pole distances $\hat{D}$. Therefore we have chosen the radii of the surface balls at a large fraction, namely at $0.95 \cdot \hat{D}(s_i)$. But the reconstructed surface nevertheless comprised holes, because still $\mathcal{F} \not\subseteq \mathcal{U}(\mathcal{B}_{\mathcal{F}})$. Another negative effect is that the pruning step is less

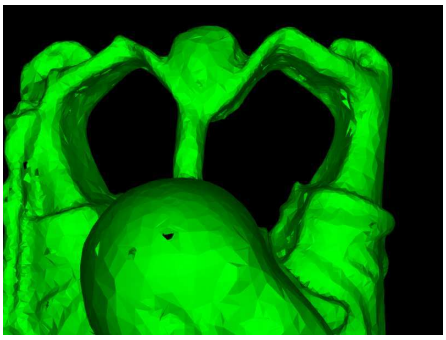
(a) Pruned set of surface balls $B'_{\mathcal{F}}$, $|B'_{\mathcal{F}}| = 92948$



(b) Surface reconstruction from $B'_{\mathcal{F}}$



(c) $B'_{\mathcal{F}}$ magnified, there are areas with unexpectedly small balls



(d) Magnified reconstructed surface with holes on top of 92948 points

Figure 4.17: Reconstruction of the *Happy Buddha* model, 543652 sample points

effective for such small surface balls, and thus $|B'_{\mathcal{F}}| = 92948$, which is quite high. We have come up with three different heuristic methods to avoid the negative effects of bad sampling quality.

1. For every ball $b_i \in B_{\mathcal{F}}$, let $s_i \in S$ be its corresponding sample point and let $s_j \in S$ denote its 3rd-nearest neighbor in S. Enforce for each ball b_i a minimum radius $\rho_i^{min} = 1.5 \cdot \|s_i - s_j\|$.

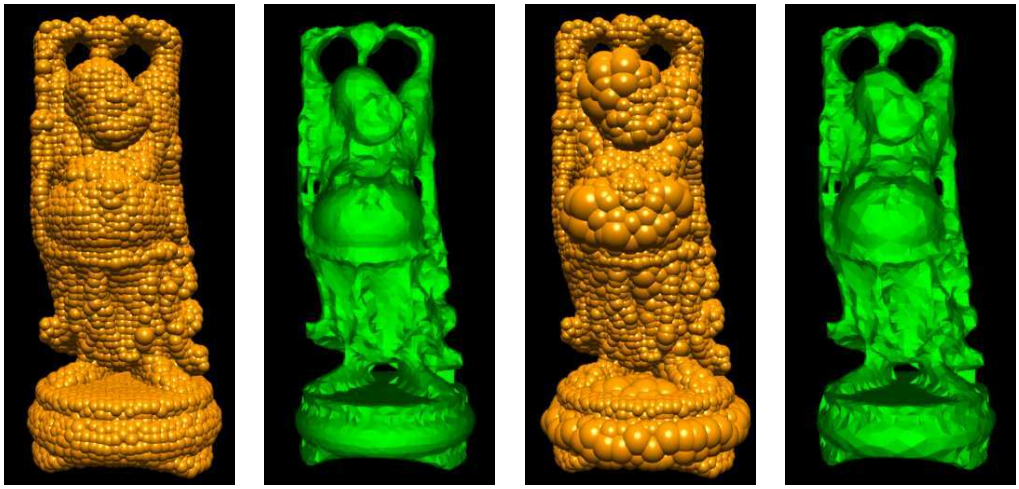
2. Same as above, but the minimum radius is globally defined by the user.

3. Compute $\hat{D}(s_i)$ not as minimum distance s_i to any pole point of DMT($\mathcal{O}$) but to any pole point of a pruned version of DMT($\mathcal{O}$) (which is more stable, see Chapter 3 for a detailed description on how to prune DMT($\mathcal{O}$)).

The second method from above sufficed to reconstruct the large *Happy Buddha* correctly. We have also tried to reconstruct this model from the dataset, which consists only of 7108 vertices, and we were again successful with a minimum radius, see Figure 4.18. For this computation, we have set the radii to $0.05 \cdot \hat{D}$ and $0.95 \cdot \hat{D}$ respectively, to illustrate again the effect of pruning. With the third heuristic from above, which estimates the local feature size using a stable medial axis transform, we may expect even better pruning. However, we have not implemented this method. Table 4.2 shows the running times for the various components of our software toolchain. Thereby the set covering step for half a million input points took more than 24 hours to complete because we have chosen best possible output quality. Note, that with less ambitious settings this task takes only a few minutes to complete. The shown *Happy Buddha* meshes contain some edges, where more than two triangles meet. This is an artefact of the weighted zero α-shape, and can be removed, but this is non-trivial from an implementation point of view. As not essential for our scientific work, we have refrained from dealing with this implementation issue.

4.10 Conclusions

In contrast to traditional surface reconstruction approaches, our method is able to simplify the given input before a mesh is computed. The observed running times are practical for moderately large data sets, but naturally cannot compete with mesh reconstruction methods that do not come with a topological guarantee (see e.g. [KBH06]). Still, our approach compares well with mesh reconstruction methods with guarantee;



(a) Radius factor 0.05, additionally a minimum radius was enforced, $|B'_{\mathcal{F}}| = 5853$

(b) Reconstructed mesh on top of 5853 vertices

(c) Radius factor 0.95, additionally a minimum radius was enforced, $|B'_{\mathcal{F}}| = 4252$

(d) Reconstructed mesh on top of 4252 vertices

Figure 4.18: Reconstruction of the *Happy Buddha* model, 7108 sample points

$\|S\|$	SB	Factor	MTX	Set cov.	$\|B'_{\mathcal{F}}\|$	Seed
[balls]	[s]		[s]	[s]	[balls]	[s]
543652	848	0.05	106.11	19988.4	319654	318
543652	848	0.95	219.86	88307.9	92948	108
7108	44.15	0.05	1.02	0.47	5853	5.4
7108	44.15	0.95	1.19	1.94	4252	4.1

Table 4.2: Running times for the reconstruction of the *Happy Buddha* dataset at different resolutions. Column "SB" shows the time for surface ball generation, "Factor" is the fraction of $\hat{D}$, which has been used for the radii, column "MTX" shows the time for computing the covering matrix, and column "Seed" contains the time consumption to extract the mesh from $B'_{\mathcal{F}}$.

see e.g. [DGH01]. The strength of our method lies in combining topological correctness with the possibility to choose the desired accuracy of the output mesh.

A limitation of our approach is its dependence on r-sampled inputs, and therefore also sensitivity to noise. However, in practical situations, heuristic methods can be used to overcome problems that originate from poor quality of the input point sample.

Chapter 5

Minkowski Sums

5.1 Motivation

Will that huge piano fit through the narrow stairway? Which paths can our robot take through the obstacles? How will the workpiece look like, when the cutting tool has moved along the planned trajectory?

Questions like these are actually questions of motion planning, offset computation and collision detection, and they are closely related to Minkowski sums. Minkowski sums of non-convex objects in $\mathbb{R}^3$ are inherently hard to compute, and often approximate results will suffice. The Minkowski sum of two balls is easy to compute, loosely speaking it is just the sum of the centers and the radii of the balls. We use this fact in our novel algorithm, to compute Minkowski sums not directly from the input objects but from their ball representations, which we get from triangular mesh representations with the method described in Chapter 3. The output of our algorithm consists of a set of balls, whose union is the Minkowski sum of the ball representations of the input objects, and it approximates the Minkowski sum of the original objects. In general, not all possible pairs of balls contribute to the union of balls that forms the approximate Minkowski sum, and we use the power diagram for a computationally inexpensive contribution test. We have explored a hierarchical approach to reduce the number of balls to consider. Techniques exist that, if required by an application, turn the boundary of the union of the computed balls back into a triangular mesh, [Kru09].

5.2 Introduction

Minkowski sums in two and three dimensions are used in various fields, for example mathematical morphology, computer graphics, convex geometry and computational geometry. Moreover they are relevant in applications such as NC machining and generalized offset computation.

This motivated many researchers to study algorithms for Minkowski sum computation. Given two sets A and B in $\mathbb{R}^2$ or $\mathbb{R}^3$, the set

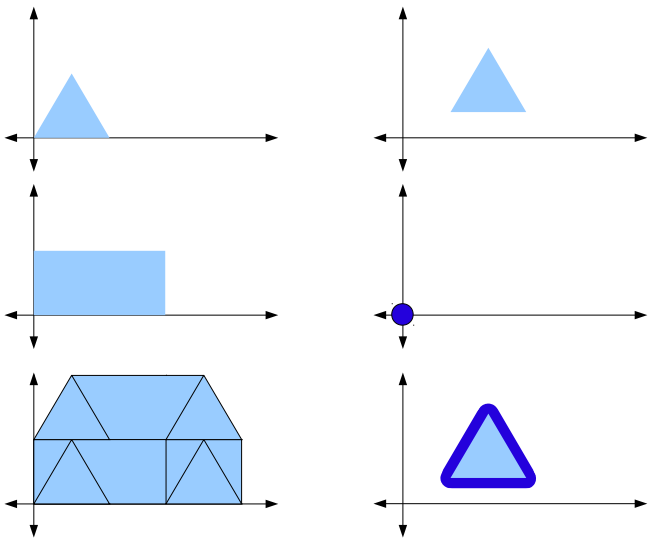
$$A \oplus B = \{a + b | a \in A, b \in B\}$$

is called the *Minkowski sum* of the sets A and B, see Figure 5.1(a). It is obvious that the sets A and B can be exchanged.

Considering the case where B is a ball of radius r, centered at the origin, the Minkowski sum $A \oplus B$ is the offset object of A at distance r, see Figure 5.1(b). Thus, at least for centrally symmetric convex sets B the Minkowski sum $A \oplus B$ can be considered as generalized offset object of A. Considering a cutting tool performing a translational motion defined by a path curve, the shape of the cut in the material is the Minkowski sum of the tool and the path. Minkowski sums occur also in testing collisions when objects undergo translational motions.

The computation of the Minkowski sum of convex solid objects A and B in $\mathbb{R}^2$ and $\mathbb{R}^3$ is a well explored subject, see [O'R98], and it is known that $A \oplus B$ is convex again. Moreover, the boundary $\partial(A \oplus B)$ of the Minkowski sum $A \oplus B$ equals the sum $\partial A \oplus \partial B$ of the boundaries ∂A and ∂B for convex input data. For non-convex input objects this is no longer true, and one is facing several difficulties. Considering volumetric objects one strategy is to perform a convex decomposition to the input objects, see e.g. [Hal02]. This results in the representations $A = \cup_i A_i$ and $B = \cup_j B_j$ with convex sets A_i and B_j which decompose the input sets A and B. The Minkowski sum $A \oplus B$ of two non-convex objects is obtained by computing the partial Minkowski sums $A_i \oplus B_j$ of all pairs of convex parts. The union of these partial sums represents the Minkowski sum.

In [VM04] this concept is applied to the Minkowski sum computation of complex polyhedral objects. The challenging part is, on the one hand, the complexity when all pairs of partial Minkowski sums are computed, and, at the other, the merging of all partial Minkowski sums. In particular, if we consider free form objects with non convex parts the convex decomposition results in a large number of parts depending on the accuracy of the computation we want to achieve.



(a) The Minkowski sum of two basic objects

(b) Offset object of a triangle, dilated by a disc

Figure 5.1: Simple Minkowski sums in 2D

We present a new method for the computation of Minkowski sums which is based on approximate representations of the input objects A and B by the unions $U(\mathcal{A})$ and $U(\mathcal{B})$ of sets of balls. We construct such sets of balls $\mathcal{A}$ and $\mathcal{B}$ with the method described in Chapter 3. We write (x, y, z, r) for a ball with center (x, y, z) and radius r. The Minkowski sum $a \oplus b$ of two balls $a = (x_a, y_a, z_a, r_a)$ and $b = (x_b, y_b, z_b, r_b)$ is again a ball, and is computationally cheap:

$$a \oplus b = (x_a + x_b, y_a + y_b, z_a + z_b, r_a + r_b)$$

Let $\mathcal{A}$ and $\mathcal{B}$ be two sets of balls which approximate the input objects A and B. In our approach, the Minkowski sum $A \oplus B$ is approximated by the Minkowski sum of the unions of these approximate ball representations. It consists of the union $U(\mathcal{A} \oplus \mathcal{B})$ of the Minkowski sums of all possible pairs of balls

$$U(\mathcal{A} \oplus \mathcal{B}) = U(a + b | a \in \mathcal{A}, b \in \mathcal{B})$$

Let $h(X_1, X_2)$ denote the one sided Hausdorff distance from a compact set X_1 to another one X_2. If $A \subseteq U(\mathcal{A})$ and $B \subseteq U(\mathcal{B})$, then

$$h(U(\mathcal{A} \oplus \mathcal{B}), A \oplus B) \leq h(\mathcal{A}, A) + h(\mathcal{B}, B)$$

holds. In many cases not all of the $|\mathcal{A}| \cdot |\mathcal{B}|$ partial Minkowski sums contribute to $U(\mathcal{A} \oplus \mathcal{B})$ and we present methods to reduce the number of partial sums.
The outline of this Chapter is as follows:

- Section 5.3 discusses the two principal strategies that we use to compute ball Minkowski sums. A simple quadratic approach is discussed as well as a hierarchical approach, based on spatial subdivison.
- Section 5.4 covers our implementation for the 2D case. This software, *amink_2d*, served as a prototype for the 3D case to gain experience with the practical aspects of our algorithms like feasibility, time and memory consumption.
- Our second implementation, *amink_3d*, deals with the 3D case. It is based on our experience with *amink_2d* and is discussed in Section 5.5, where it is also compared to existing approaches.
- Finally, in Section 5.6 we conclude our work.

5.3 Minkowski sums of unions of balls

The union of the set of balls $\{a \oplus b | a \in \mathcal{A}, b \in \mathcal{B}\}$ is the Minkowski sum of $U(\mathcal{A})$ and $U(\mathcal{B})$. But generally, not all of these balls contribute to their union. In this section we describe an efficient contribution test, and we describe two approaches to compute the minimal set $\mathcal{C} \subset \mathcal{A} \oplus \mathcal{B}$ such that $U(\mathcal{C}) = U(\mathcal{A} \oplus \mathcal{B})$. This set $\mathcal{C}$ then is the output of our algorithms.

5.3.1 The quadratic approach

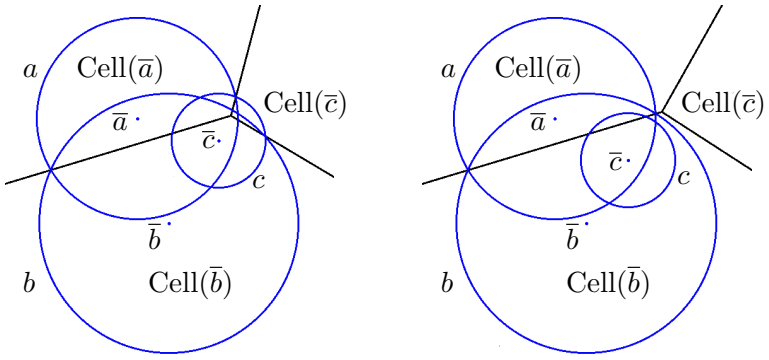
A simple quadratic approach tentatively computes $\mathcal{C}$ as the set of all $|\mathcal{A}| \cdot |\mathcal{B}|$ partial Minkowski sums. Then it discards those balls from $\mathcal{C}$ which are redundant, i.e., do not contribute to $U(\mathcal{C})$. To find out if a ball $c \in \mathcal{C}$ contributes to $U(\mathcal{C})$, let $\overline{\mathcal{C}}$ be a set of weighted points, consisting of the centers of the balls in the set C, weighted by their squared radii, i.e.

$$\overline{\mathcal{C}} = \{(c_x, c_y, c_z, c_r^2) | (c_x, c_y, c_z, c_r) \in \mathcal{C}\}$$

We compute the power diagram $P(\overline{\mathcal{C}})$, definied in Section 2.4. Let $\mathrm{Cell}(\overline{c})$ denote the power cell of a weighted point $\overline{c} \in \overline{\mathcal{C}}$ corresponding to a ball $c \in \mathcal{C}$. We decide weather to keep c or not, using the property

$$U(\mathcal{C}) \neq U(\mathcal{C} \setminus c) \Leftrightarrow \mathrm{Cell}(\overline{c}) \cap c \neq \emptyset. \tag{5.1}$$

That is, c contributes to $U(\mathcal{C})$ iff the intersection of c with its own power cell Cell($\overline{c}$) is non-empty [Aur88, Lemma 1], see Figure 5.2.



(a) Each ball contributes with those parts of its boundary to the union of the three balls which lie within the power cell of its corresponding weighted point

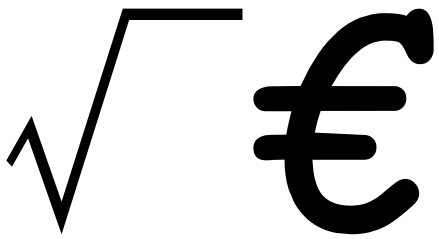
(b) Only the balls a and b contribute to the union of the three balls, because c doesn't intersect the power cell of its corresponding weighted point $\overline{c}$

Figure 5.2: Three balls and their power diagram

5.3.2 The hierarchical approach

The described approach, namely constructing $a_i \oplus b_j$ for each possible pair and deleting the balls that do not contribute to $U(\mathcal{C})$, works correctly and is worst case optimal. However, in cases where a large fraction of these balls does not contribute, a hierarchical approach might be better suited. We have developed such an approach, and, depending on the input, it often considerably reduces the number of ball pairs to be processed.

W.l.o.g., assume that $|\mathcal{B}| \geq |\mathcal{A}|$. We generate for $\mathcal{B}$ an octree-like data structure $T_{\mathcal{B}}$, where each node N_β in $T_{\mathcal{B}}$ represents the subset $\beta \subseteq \mathcal{B}$ of balls having their centers in the octree cell associated with N_β. In each node we store the smallest enclosing circumball S_β of $U(\beta)$. The leafs of $T_{\mathcal{B}}$ are buckets, containing a small number of balls of $\mathcal{B}$. The hierarchical Minkowski sum algorithm takes $\mathcal{A}$ and $T_{\mathcal{B}}$ as input and computes the result $\mathcal{C}$ as well as $P(\overline{\mathcal{C}})$ iteratively. It traverses $T_{\mathcal{B}}$ for every ball $a \in \mathcal{A}$ in a top-down manner and checks if the ball $a \oplus S_\beta$ would contribute to $U(\mathcal{C})$. If this is not the case, the set $\{a \oplus b | b \in \beta\}$ can not contribute to $U(\mathcal{C})$ and we can stop the computations for a in the respective branch of $T_{\mathcal{B}}$. Otherwise the (up to) eight sons of the current node N_β of $T_{\mathcal{B}}$ must be traversed for a in a recursive manner. Whenever



(a) Polygonal model *sqroot*, 10 edges

(b) Polygonal model *euro*, 363 edges

Figure 5.3: Input objects in 2D

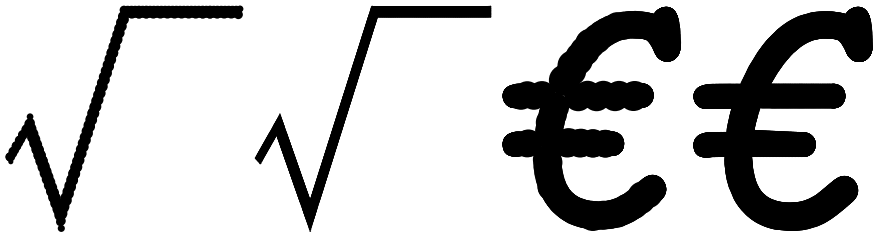
a leaf, i.e.; a bucket, is reached, the contained balls are checked for contribution. The ones that contribute are added to $\mathcal{C}$, and $P(\overline{\mathcal{C}})$ is updated. $\overline{\mathcal{C}}$ is only partially known during the algorithm, and so some balls which do not contribute to $U(\mathcal{C})$ might survive the contribution tests. They are removed afterwards in a cleanup step using the already existing power diagram $P(\overline{\mathcal{C}})$ and the property from Equation 5.1.

5.4 2D Minkowski sums

We use the models *sqroot* and *euro*, shown in Figure 5.3, to evaluate our software *amink_2d*. The input for *amink_2d* has to consist of sets of discs, so we have preprocessed the two polygons with a 2D version of the polyhedron-to-balls conversion method described in Chapter 3. We refrain from repeating the details here as the algorithm is straightforward for one dimension less, and as its time consumption is negligible. We have approximated each of the two polygonal models by a coarse and a more accurate union of discs. Thereby, the new representation of the *euro* model consists of 321 and 486 discs, and the *sqroot* model is represented by 157 and 565 discs, respectively, see Figure 5.4.

We denote by *algo_sq* the quadratic algorithm (Section 5.3.1) where the Minkowski sum of each possible pair of balls is computed and tested for contribution. The more sophisticated hierarchical approach is referred to as *algo_h*. The two approaches are implemented in our *amink_2d* software, and Table 5.1 shows their running times for the coarsely approximated inputs as well as for the more accurate version. The resulting approximate Minkowski sums are shown in Figure 5.5(a) and in Figure 5.5(b).

To compare our approximate results with the real Minkowski sum of the original



(a) Model *sqroot*, 157 discs

(b) Model *sqroot*, 565 discs

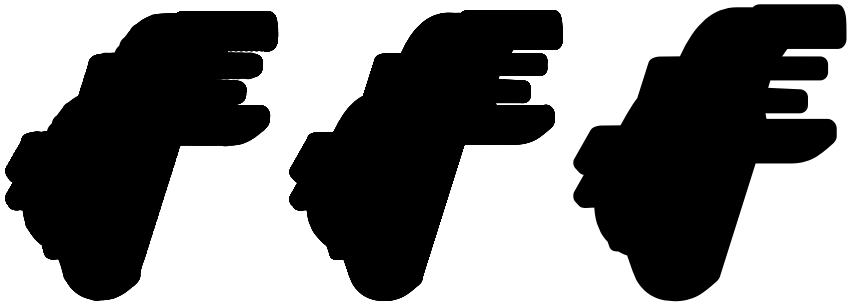
(c) Model *euro*, 321 discs

(d) Model *euro*, 486 discs

Figure 5.4: Input objects, represented by unions of discs

sqroot	*euro*	$\lvert sqroot \rvert \cdot \lvert euro \rvert$	$\lvert\mathcal{C}\rvert$	*algo_sq*	*algo_h*
[discs]	[discs]	[discs]	[discs]	[s]	[s]
157	321	50397	7688	10.20	16.61
565	486	274590	43165	185.36	338.38

Table 5.1: Running times of *amink_2d*, benchmarked on a Intel Core i7 940 CPU running at 3 GHz



(a) Minkowski sum, 7688 discs

(b) Minkowski sum, 43165 discs

(c) Exact Minkowski sum of the polygonal models *sqroot* and *euro*

Figure 5.5: Minkowski sums of the disc representations as compared to the exact Minkowski sum of the original polygonal models

input objects we have computed *sqroot* $\oplus$ *euro* with the exact *Minkowski_sum_2* package [Wei09] from CGAL. The output is shown in Figure 5.5(c). The *Minkowski_sum_2* package showed an execution time of less than 100 ms for this task, i.e. it greatly outperforms *amink_2d*. As the package is also amazingly fast for more complicated inputs, there is probably no need for our approach in 2D (at least not with respect to running time). We nevertheless analyze the reasons for the long running times of *amink_2d* in the next section.

5.4.1 Analysis of *amink_2d*

We consider the example where the inputs were represented by 565 and 486 balls, respectively. Here our hierarchical approach *algo_h* avoided the computation of 29 percent of those partial Minkowski sums that are not part of the final solution. But although *algo_h* is apparently superior to *algo_sq*, it showed a worse running time. The reason for this behavior can be found in the way we have implemented *amink_2d*, which doesn't account for the way the underlying regular triangulation package from CGAL works.

Let T denote an instance of CGAL's regular triangulation class [Yvi09]. As described in Section 5.3.2, our algorithm *algo_h* frequently tests if the Minkowski sum of a disc $a \in \mathcal{A}$ and the circumdisc S_β of a set of discs $\beta \in \mathcal{B}$ contributes to the Minkowski sum computed so far. For this test, a weighted point $\overline{S_\beta}$, corresponding to S_β, is inserted into the regular triangulation T computed so far. Then the contribution test is performed, and then $\overline{S_\beta}$ is immediately removed again. This is a straightforward implementation. But when we stop treating the underlying triangulation package as black box, it becomes clear that this implementation is disadvantageous: As S_β is a circumdisc of a number of discs, the weight of $\overline{S_\beta}$ can be huge compared to the weights of the points in T. Thus, when $\overline{S_\beta}$ is inserted into T, a large set of the triangles of T is probably in conflict with this point. These triangles need to be removed, and the conflict zone needs to be re-triangulated. Even worse, when $\overline{S_\beta}$ is removed again, not only the just introduced changes are to be reverted, but also the set of the so called *hidden vertices* must be considered. This term requires an explanation: T possibly uses only a subset of its input points as vertices of the triangulation. The other ones are stored as *hidden vertices*. Whenever a point is removed from T, *hidden vertices* can re-appear in the triangulation. Thus they must all be checked when $\overline{S_\beta}$ is removed from T.

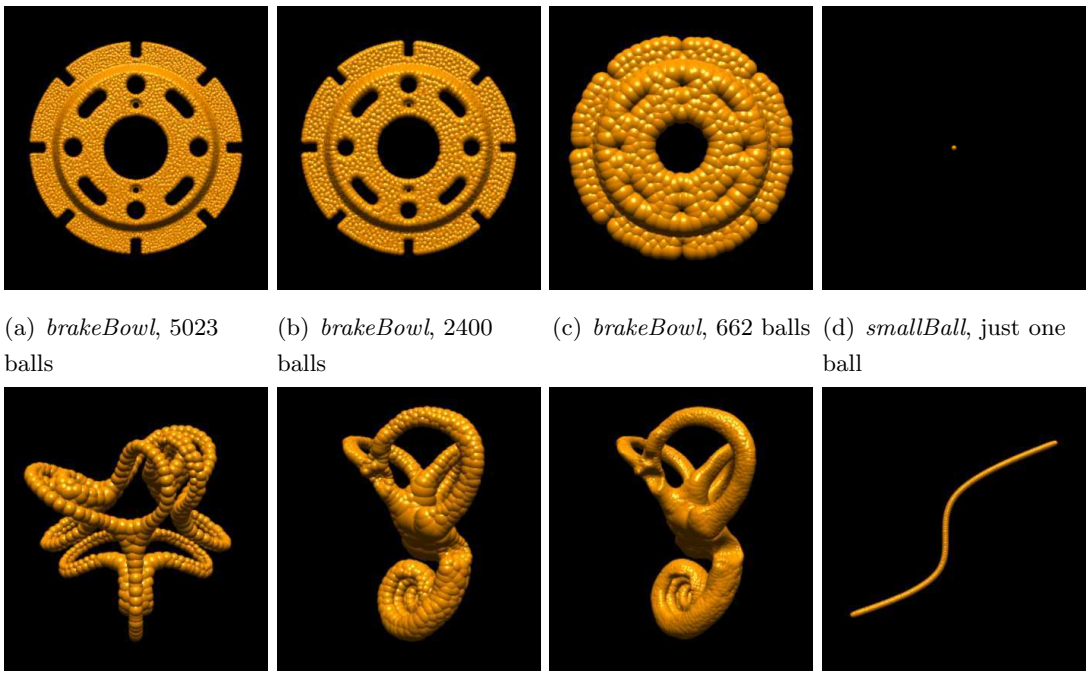
Yet another problem that affects *algo_h* is the way the underlying triangulation

library is implemented. The respective class applies an insertion algorithm, which sounds appropriate at first sight. But at each insertion, a time critical point location query is necessary, and this can considerably slow down the triangulation process. If all points can be passed to T at once, then the points can be preprocessed, i.e., CGAL first orders them along a Hilbert curve and then inserts them in that order. This order is particularly advantageous, because it provides locality, i.e., two consecutive points are with high probability nearby. Now, each time the point location routine is called, it takes the previous result as initial guess, and this way the time consumption of the point location queries becomes negligible. Moreover, this insertion order is cache friendly, i.e., temporally consecutive changes in T will occur locally, and thus there is a good chance that the affected elements of T are already cached when they are referenced. Unfortunately, the hierarchical version of our approach, *algo_h*, can only pass the input points one by one to T. Thus the advantage of this optimization is lost.

We have explored the impact of the insertion order to the running time of the bare triangulation process. When we passed a million of randomly created points at once to CGAL's Delaunay triangulation class, the triangulation is computed within a second. However, when we passed the same points one by one, the time consumption was 104 seconds. We must nevertheless not conclude that random insertion order is generally a hundred times worse. Recently I have written my own 2D triangulation class, and I have observed a much better running time for random order insertion (at the price of a slightly higher memory consumption for a point location data structure). As *amink_2d* serves just as a prototype for the 3D case, we have not ported it to use the new class. But we believe that the hierarchical approach is still attractive, if it is properly supported by the underlying triangulation class.

5.5 3D Minkowski sums

Our implementation for the 3D case is called *amink_3d*, and it has been developed based on the experience we gained with *amink_2d*. Let T again denote an instance of a regular triangulation. As to the hierarchical approach: We would have been able to test circumballs for contribution without actually inserting them into T. But for optimal running time and memory consumption we demand proper support for random order point insertion, and it should be possible to just discard hidden vertices. This is currently not possible with CGAL, so we have instead implemented the quadratic algorithm (Section 5.3.1) in two different ways. We refer to them as *algo_mem* and



(a) *brakeBowl*, 5023 balls (b) *brakeBowl*, 2400 balls (c) *brakeBowl*, 662 balls (d) *smallBall*, just one ball

(e) *knot*, 682 balls (f) *innerEar*, 687 balls (g) *innerEar*, 2915 balls (h) *s-tube*, 135 balls

Figure 5.6: Input objects, represented by unions of balls

algo_time. Thereby, *algo_mem* inserts a weighted point $\overline{c}$ for the Minkowski sum c of every possible pair of balls into T and immediately checks if c contributes to the union of the so far computed partial Minkowski sums. If c doesn't contribute, $\overline{c}$ is removed immediately from T. Thus, *algo_mem* is memory efficient. However, a post-processing step is necessary: As not all partial Minkowski sums are known at the time of the contribution tests, the surviving balls must be re-checked again at the end. Further, *algo_mem* can only pass the weighted points one by one to T, and in conjunction with CGAL this slows down the triangulation process as described in the previous section.

Our second algorithm, *algo_time*, first computes the Minkowski sums of all possible pairs of balls. Then it passes the corresponding weighted points at once to T. The contribution checks are performed only once and only for the balls that do not correspond to hidden vertices of the triangulation. Obviously, *algo_time* performs faster, but at the price of a higher memory consumption.

5.5.1 3D examples

We use the models shown in Figure 5.6 to evaluate our approach, and in order to compare it with existing approaches, [Lie08, Hac09], for which software is publicly available. We have partially downloaded these models from [Jyh, AIM] and have converted them into unions of balls using the method described in Chapter 3.

Our first example combines $\mathcal{A} = \textit{brakeBowl}$ with the $\mathcal{B} = \textit{smallBall}$ model, which consists just of one ball. The resulting object is a special case of Minkowski sum, namely it is the offset object of *brakeBowl* for the offset r, i.e., this offset object consists of all points of $\mathbb{R}^3$ that lie within a distance r from *brakeBowl*, where r is the radius of *smallBall*. We have created the *brakeBowl* model at three different levels of accuracy, see Figure 5.6(a), 5.6(b), and 5.6(c), and the computed offset objects $\mathcal{A} \oplus \mathcal{B}$ are shown in Figure 5.7(a), 5.7(b), and 5.7(c). Table 5.2 shows the corresponding running times and memory consumptions and compares them to the ones of CGAL's *Minkowski_sum_3* package [Hac09], shortly referred to as *mcgal* in the following, and to the ones of the software *m+3d* [Lie08, Jyh].

Our software computed the result within 21 seconds, and its memory consumption stayed below 8 MB as opposed to the exact software *mcgal*, which needed 2 hours and almost 11 GB main memory. The other approximate software, *m+3d*, threw an assertion:

```
"src/minkowski-facet-stitch.h:103: m_f_graph_face* get_valid_face(
    m_f_graph*, uint): Assertion '0' failed".
```

We have written a bug report, and the authors of the software have kindly promised to have a look at the problem. However, at the time we finished this work, the reason why the assertion was triggered was still unclear. In our second example we simulate the movement of an object along a path, namely we set $\mathcal{A} = \textit{innerEar}$ and $\mathcal{B} = \textit{s-tube}$. Here, a considerable fraction of the $|\mathcal{A}| \cdot |\mathcal{B}|$ balls contributes to $U(\mathcal{A} \oplus \mathcal{B})$ (shown in Figure 5.7(d)), and our approach needed 202 seconds. The software *m+3d* was apparently able to benefit from the particularly simple structure of the *s-tube* model, and it finished successfully within less than 14 seconds. The exact software *mcgal*, however, threw an assertion:

```
terminate called after throwing an instance of 'CGAL::
    Assertion_exception'
  what():  CGAL ERROR: assertion violation!
File: /opt/CGAL351/include/CGAL/Convex_decomposition_3/Ray_hit_generator
    .h
Line: 152
```

```
Explanation: ray should hit vertex, edge, or facet
Aborted
```

We have made a detailed bug report, but we got no answer from the CGAL project. Therefore we can't interpret what happened exactly.

For our third example we have set $\mathcal{A} = innerEar$ and $\mathcal{B} = knot$, and with our software we have computed the Minkowski sum for two different levels of accuracy, shown in the Figures 5.7(e) and Figure 5.7(f). As our approach is not affected by the complicated structure of these examples, it has successfully accomplished this task within 28 and 201 seconds, respectively. The software *m+3d* took 825 seconds, and it had a worse memory consumption. The *mcgal* software terminated again with an assertion.

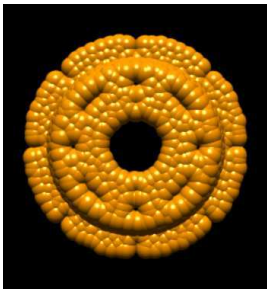
Actually, we also intended to compare our approach with the topology retaining approximate approach [VM04], but unfortunately the authors do not have the models they have used for their paper anymore. We have also made a detailed bug report about the bug that occured in *mcgal*, but got no response from the CGAL project. So we settle with the results of our software *amink_3d* and the partial results of *mcgal* and *m+3d*. Clearly, the comparability of the three different approaches is limited: One is exact, the other two are not. Ours is robust and it has to pay penalty for the use of filtered arithmetic (see Section 2.1.2), while the robustness issues with *m+3d* might come from the use of fast and inexact predicates. Also, one would actually need to put the accuracies into perspective, by which the different representations approximate the real input objects. But then still the shape of the input objects would heavily influence which algorithm is the best choice.

5.6 Conclusions

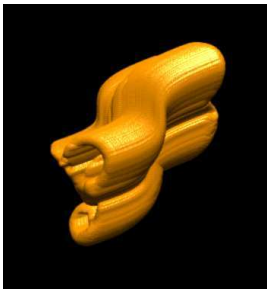
We have developed and implemented a new approach for approximate Minkowski sum computations. We have successfully applied our implementation to different models in settings like motion planning and offset computation. We have also compared our results with the exact Minkowski sum package from CGAL as well as with the publicly available *m+3d* software for approximate Minkowski sums. As our approach operates on unions of balls it is not directly comparable to any existing one. But our examples show that our software scores well with respect to robustness, running time and memory consumption. Our approach does not guarantee that the topology of the approximate Minkowski sum equals the one of the exact Minkowski sum of the input objects. But

(a) Offset computation: *brakeBowl* $\oplus$ *smallBall*, $5023 \oplus 1 = 5023$ balls

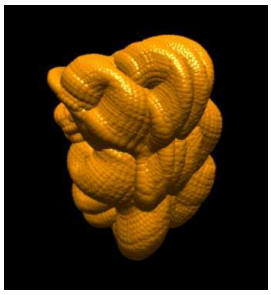
(b) Offset computation: *brakeBowl* $\oplus$ *smallBall*, $2400 \oplus 1 = 2400$ balls



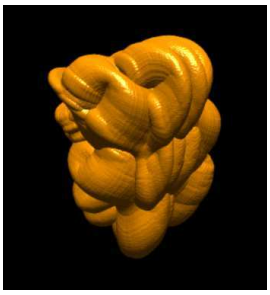
(c) Offset computation: *brakeBowl* $\oplus$ *smallBall*, $662 \oplus 1 = 662$ balls



(d) Moving an object along an s-shaped path: *innerEar* $\oplus$ *s-tube*, $2915 \oplus 135 = 167692$ balls



(e) *innerEar* $\oplus$ *knot*, $687 \oplus 682 = 63143$ balls



(f) *innerEar* $\oplus$ *knot*, $2915 \oplus 1388 = 273112$ balls

Figure 5.7: Minkowski sums

Model 1 $\lvert\mathcal{A}\rvert$, $\lvert A\rvert$	Model 2 $\lvert\mathcal{B}\rvert$, $\lvert B\rvert$	$\lvert\mathcal{A}\rvert \cdot \lvert\mathcal{B}\rvert$ [#balls]	|M. sum| [#balls]	*algo_time*	*algo_mem*	*mcgal*	*m+3d*
brakeBowl 5023,7532	smallBall 1,501	5023	5023	21.22 s 8.16 MB	21.32 s 7.52 MB	7267.87 s 10,96 GB	•
brakeBowl 2400,7532	smallBall 1,501	2400	2400	11.29 s 5.109 MB	11.29 s 4.80 MB	7267.87 s 10,96 GB	•
brakeBowl 662,7532	smallBall 1,501	662	662	2.8 s 2.94 MB	2.8 s 2.94 MB	7267.87 s 10,96 GB	•
innerEar 2915,32236	s-tube 135,500	393525	167692	201.74 s 304.44 MB	228.72 s 252.64 MB	•	13.53 s 144 MB
innerEar 687,32236	knot 682,992	468534	63143	28.18 s 269.84 MB	39.76 s 198.91 MB	•	825 s 2.14 GB
innerEar 2915,32236	knot 1388,992	4046020	273112	201.34 s 2.18 GB	387.34 s 1.53 GB	•	825 s 2.14 GB

Table 5.2: Running times and memory consumption for the 3D case on an Intel Core i7 940 CPU running at 3 GHz with 12 GB RAM. The first and second columns show the input models and the number of balls and triangles by which they are represented. At "•" entries, the respective software with which we compare ours did not succeed.

in fact, we know only of one approximation approach [VM04] which comes with such a guarantee, and this approach, as well as the exact Minkowski sum software of CGAL are limited to relatively small inputs. Thus one might need to simplify real world objects and thereby lose the topological guarantee anyway. In view of this, the lack of topological guarantees is not a real drawback of our approach. Still, the choice of the best approach depends heavily on the shape of the input objects as well as on the demands on the output quality.

Chapter 6

Conclusions

6.1 Abstract

Algorithmic solutions for three quite different problems have been considered independently from each other in the previous chapters. In this chapter we establish connections between them. For this purpose we extract the principal functional blocks of our approaches. We show that many of these functional blocks are shared between our algorithms, and we demonstrate how everything fits in a big framework. We discuss further applications of our techniques, and we also discuss its limitations.

6.2 The common framework

Figure 6.1 shows the overall framework that has evolved. All algorithms start with an r-sample of one or more objects, and they terminate with their respective construction, i.e., with a seed polytope, a reconstructed surface, an approximate medial axis, an approximate Minkowski sum of two objects, or with a *shape prior* for medical image segmentation. The latter has not been introduced yet because it is not a subject of this thesis. A short description of our related work is given in Section 6.3.1.

6.2.1 Computation of the discrete medial axis transform

The topmost functional block in the diagram in Figure 6.1 is common to all algorithms. It computes from a given r-sample S of an object $\mathcal{O}$ a discrete medial axis transform $\mathrm{DMT}(\mathcal{O})$. This set of balls serves two purposes. Firstly, it supports estimation of the local feature sizes in S, which is required for the construction of proper surface balls. Secondly, the union of $\mathrm{DMT}(\mathcal{O})$ approximates $\mathcal{O}$, and this property is used by the other algorithms.

6.2.2 Pruning by set covering

The set covering block in Figure 6.1 is a central element of our framework, because all algorithms introduced in this thesis involve pruning. The motivation to prune sets of balls is different in the various approaches. In the case where surface balls $B_{\mathcal{F}}$ are treated, density reduction is the primary goal. Therefore, this is the stage where the degree of accuracy for subsequent surface reconstruction from $B'_{\mathcal{F}}$ is controlled. In the other case, where approximate medial balls $\mathrm{DMT}^{+}_{\mathrm{in}}$ are treated, density reduction is still very important. But in this case, the key goal is stabilization, i.e., removal of balls which correspond to 'nonrelevant' features of $\mathcal{O}$.

No matter what kind of balls is supplied, the set covering algorithm treats them as sets in a mathematical sense, i.e., it operates purely combinatorially. Therefore, this step must be adapted to the kind of input to maintain topological correctness, as described in the respective chapters.

6.2.3 The weighted zero α-shape

In the surface reconstruction branch of our framework, we compute the weighted zero α-shape $\mathcal{A}(B'_{\mathcal{F}})$ of a pruned set of surface balls $B'_{\mathcal{F}}$, and in a subsequent step a

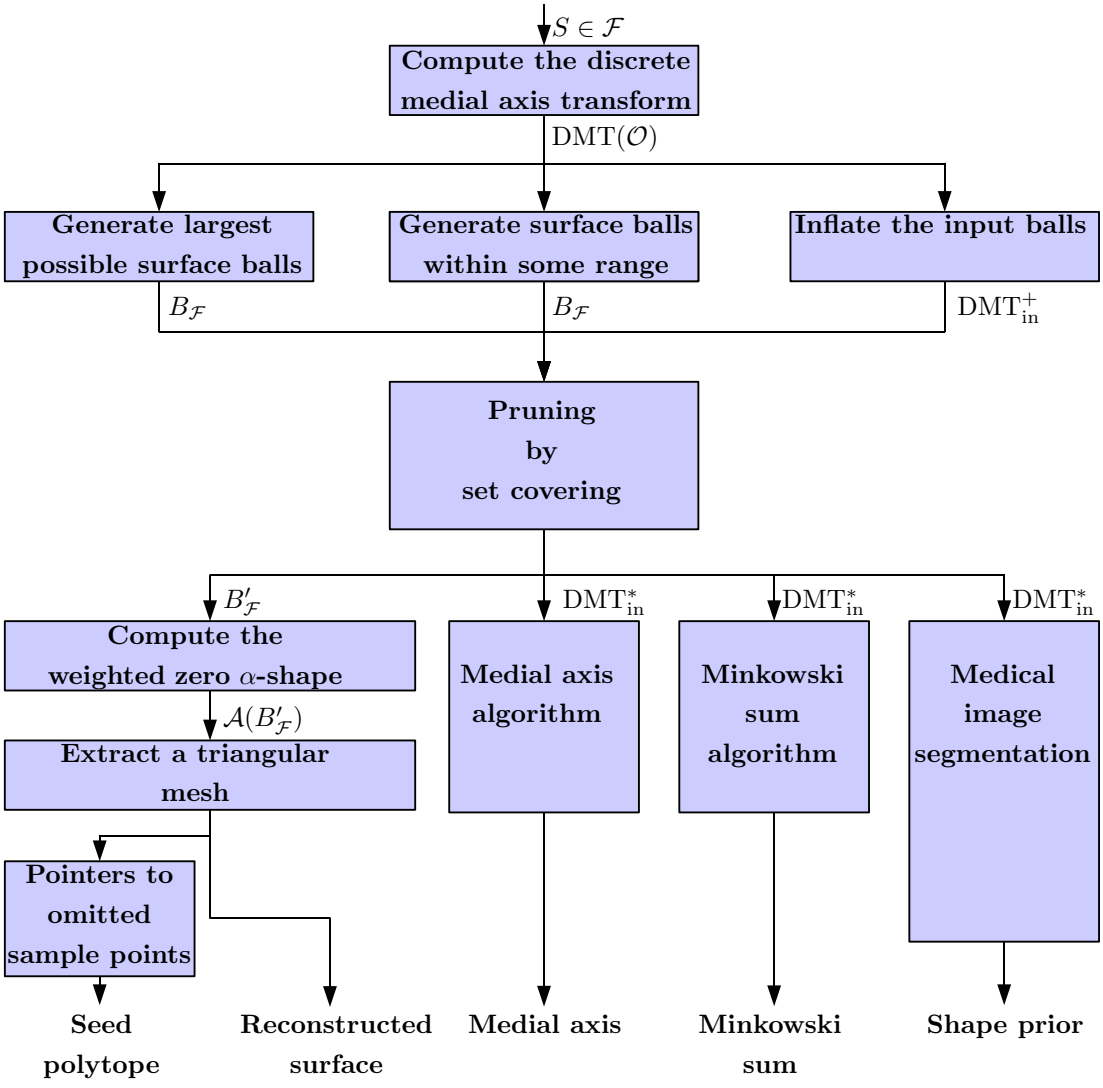


Figure 6.1: Overall framework

triangular mesh is extracted from $\mathcal{A}(B'_{\mathcal{F}})$, which serves either as a reconstructed surface or as part of the seed polytope. Note that we discuss local feature sizes in Chapter 4, using the weighted zero α-shape $\mathcal{A}(\mathrm{DMT}(\mathcal{O}))$ of $\mathrm{DMT}(\mathcal{O})$. But $\mathcal{A}(\mathrm{DMT}(\mathcal{O}))$ is only virtually used there, and thus $\mathcal{A}(\mathrm{DMT}(\mathcal{O}))$ is not part of the framework.

6.3 Further applications

Over the last years, a wide range of different people was interested in our constructions. Among them were scientists from various fields of research, an arms manufacturer (we did not collaborate here), a video game producer as well as a 'Voronoi' sculptor. Their interest provides evidence that our constructions can be applied to many real-world problems. One of these problems, whose connection to our constructions is not obvious at first sight, is medical image segmentation. Jointly with scientists from this field of research, we have developed a new segmentation approach, as desribed in the following.

6.3.1 Medical image segmentation

Medical image segmentation is a key to success of better medial imaging. But missing data, low contrast and noise, often present in medical images, impede the segmentation of objects of interest. In [JAS] we tackle the problem with a pipeline of algorithms, ranging from the ones in our framework to statistical methods.

Human organs, that have been segmented from medical image data by experts, serve as training data for our approach. From the supplied meshes, we compute with our framework small sets of balls whose unions approximate the organs in a stable way. Then, after an alignment and matching step, a statistical model is established from the balls, which assists later in semi-automatic segmentation of such organs. We refer the interested reader to [JAS] and [Abh10].

6.4 Limitations

A limitation of our approach is that thin or very detailed objects are not well suited to approximation by balls. Consider, for example, car body parts like an engine hood made of metal sheet. A large, memory consuming set of tiny balls is required to approximate such an object. An idea to avoid excessive memory consumption is to base the approximation on ellipsoids (see, for example, [BK02]). This does not require a new framework, because scaling the original object in its principal directions and

treating it by balls equals treatment of the original object by axis-aligned ellipsoids. On the other hand, the available memory in average office computers has increased steeply over the last five years of our research. This provides evidence that memory consumption will become an even less limiting factor in the future.

Bibliography

[AAH+07] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Kornberger, M. Peternell, and H. Pottmann. Approximating boundary-triangulated objects with balls. In *Proc. 23rd European Workshop on Computational Geometry*, pages 130–133, Graz, 2007.

[AAK+09] O. Aichholzer, F. Aurenhammer, B. Kornberger, S. Plantinga, G. Rote, A. Sturm, and G. Vegter. Recovering structure from r-sampled objects. In *Proc. Symposium on Geometry Processing*, volume 28, pages 1349–1360, Berlin, 2009.

[AB99] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22:481–504, 1999.

[Abh10] Jochen Abhau. *Geometrical and Statistical Methods for Segmentation of Topologically Complex Objects in 3D.* PhD thesis, Leopold Franzens Universität Innsbruck, 2010.

[ACK01] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2001.

[AIM] Aim@shape shape repository. `http://shapes.aim-at-shape.net/`.

[AK00] N. Amenta and R. Kolluri. Accurate and efficient unions of balls. In *Proc. 16th Ann. Symp. Computational Geometry*, pages 119–128, Hong Kong, 2000. ACM.

[AK01] N. Amenta and R. Kolluri. The medial axis of a union of balls. *Computational Geometry: Theory and Applications*, 20:25–37, 2001.

[AM97] D. Attali and A. Montanvert. Computing and simplifying 2d and 3d continuous skeletons. *Computer Vision and Image Understanding*, 67:261–273, 1997.

[Aur88] F. Aurenhammer. Improved algorithms for discs and balls using power diagrams. *Journal of Algorithms*, 9:151–161, 1988.

[Aur91] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991. Habilitationsschrift. [Report B 90-09, FU Berlin, Germany, 1990].

[BBP01] Hervé Brönnimann, Christoph Burnikel, and Sylvain Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. *Discrete Appl. Math.*, 109(1-2):25–47, 2001.

[BC00] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 223–232, New York, NY, USA, 2000. ACM.

[BC01] J.-D. Boissonnat and F. Cazals. Natural neighbor oordinates for points on a surface. *Computational Geometry: Theory and Applications*, 19:155–173, 2001.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[BK02] S. Bischoff and L. Kobbelt. Ellipsoid decomposition of 3d-models. In *3DPVT02*, pages 480–488, 2002.

[BO04] Gareth Bradshaw and Carol O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph.*, 23(1):1–26, 2004.

[BPR+07] K. Buchin, S. Plantinga, G. Rote, A. Sturm, and G. Vegter. Convex approximation by spherical patches. In *Proc. 23rd European Workshop on Computational Geometry*, pages 26–29, Graz, 2007.

[BT06] J.-D. Boissonnat and Monique Teillaud, editors. *Effective Computational Geometry for Curves and Surfaces*. Springer, 2006.

[CCSL09] F. Chazals, D. Cohen-Steiner, and A. Lieutier. A sampling theory for compact sets in Euclidean spaces. *Discrete & Computational Geometry*, 41:461–479, 2009.

[CGA] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org/`, last accessed: 27. Oct. 2009.

[CL08] F. Chazal and A. Lieutier. Smooth manifold reconstruction from noisy and non-uniform approximation with guarantees. *Computational Geometry: Theory and Applications*, 40:156–170, 2008.

[CM69] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, New York, NY, USA, 1969. ACM.

[CMF93] Olivier Coudert, Jean Christophe Madre, and Henri Fraisse. A new viewpoint on two-level logic minimization. In *DAC '93: Proceedings of the 30th international Design Automation Conference*, pages 625–630, New York, NY, USA, 1993. ACM.

[DG06] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. *Comput. Geom. Theory Appl.*, 35(1):124–141, 2006.

[DGH01] T. K. Dey, J. Giesen, and J. Hudson. Delaunay based shape reconstruction from large data. In *Proc. IEEE Symp. Parallel and Large-Data Visualization and Graphics*, pages 19–27, San Diego, 2001.

[DV77] A. K. Dewdney and J. K. Vranch. A convex partition of R^3 with applications to Crum's problem and Knuth's post-office problem. *Utilitas Math.*, 12:193–199, 1977.

[Dwy89] R. A. Dwyer. Higher-dimensional voronoi diagrams in linear expected time. In *SCG '89: Proceedings of the fifth annual symposium on Computational geometry*, pages 326–333, New York, NY, USA, 1989. ACM.

[DZ04] T. Dey and W. Zhao. Approximating the medial axis from the "voronoi" diagram with a convergence guarantee. 38:179–200, 2004.

[Ede92] Herbert Edelsbrunner. Weighted alpha shapes. Technical report, Champaign, IL, USA, 1992.

[Ede95] H. Edelsbrunner. The union of balls and its dual shape. *Discrete & Computational Geometry*, 13:415–440, 1995.

[Fei96] U. Feige. A threshold of ln n for approximating set cover. Proc. 28^{th} Ann. ACM Symp. on Theory of Computing, pages 314–318, 1996.

[FHW08] Efi Fogel, Dan Halperin, and Christophe Weibel. Abstract on the exact maximum complexity of minkowski sums of convex. 2008.

[FLM03] M. Foskey, M. C. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Proc. 8th ACM Symp. Solid Modeling and Applications*, pages 96–107, 2003.

[Fre01] Free Software Foundation. GNU Lesser General Public License (GPL). `http://www.gnu.org/licenses/lgpl.html`, 2001.

[FS06] A. Fabri and S.Pion. A generic lazy evaluation scheme for exact geometric computations. In *Proc. 2nd Library-Centric Software Design*, pages 75–84, 2006.

[GCC] GCC, the GNU Compiler Collection. `http://gcc.gnu.org/`.

[GMP] GNU Multiple Precision Arithmetic Library. `http://gmplib.org/`, last accessed: 12. Nov. 2009.

[GMP07] J. Giesen, B. Miklos, and M. Pauly. Medial approximation of planar shapes from union of balls: A simpler and more robust algorithm. In *Prof. 19th Canad. Conf. Comput. Geom. (CCCG)*, pages 105–108, 2007.

[GO08] L. J. Guibas and S. Y. Oudot. Reconstruction using witness complexes. *Discrete & Computational Geometry*, 40(3):325–356, 2008.

[Hac09] Peter Hachenberger. 3D Minkowski sum of polyhedra. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.5 edition, 2009.

[Hal02] D. Halperin. Robust geometric computing in motion. *International Journal of Robotics Research*, 21(3):219–232, 2002.

[Hub96] P. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15:179–210, 1996.

[JAS] Sebastian Colutto Jochen Abhau, Oswin Aichholzer and Otmar Scherzer. Shape spaces via medial axis transforms for segmentation of 3d voxel data. Submitted to the *European Journal of Applied Mathematics* in January 2010.

[Jyh] Jyh-Ming Lien. George Mason University. `http://cs.gmu.edu/~jmlien/masc/index.php?n=Main.SimpleMKSum`, last accessed: Jan. 2010.

[Kar72] R.M. Karp. Reducability among combinatorial problems. Plenum Press, New York, pages 85–103, 1972.

[KBH06] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Eurographis Symposium on Geometry Processing*, pages 61–70, 2006.

[Kru09] Nico Kruithof. 3D skin surface meshing. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.5 edition, 2009.

[Lie08] Jyh-Ming Lien. A simple method for computing minkowski sum boundary in 3d using collision detection. In *Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2008.

[LPS] lp_solve, a mixed integer linear programming (milp) solver. Available online at `http://lpsolve.sourceforge.net/`, last accessed: 27. Oct. 2009.

[NSW08] P. Niyogi, S. Smale, and S. Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39:419–441, 2008.

[O'R98] Joseph O'Rourke. *Computational geometry in C (2nd ed.)*. Cambridge University Press, New York, NY, USA, 1998.

[Pap76] Christos H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16(3):263–270, 1976.

[RF96] V. Ranjan and A. Fournier. Matching and interpolation of shapes using unions of circles. *Computer Graphics Forum*, 15:129–142, 1996.

[SAAY06] M. Samozino, M. Alexa, P. Alliez, and M. Yvinec. Reconstruction with Voronoi centered radial basis functions. In *Proc. 4th Eurographics Symp. on Geometry Processing*, pages 51–60, 2006.

[SB98] Doron Shaked and Alfred M. Bruckstein. Pruning medial axes. *Comput. Vis. Image Underst.*, 69(2):156–169, 1998.

[SFM07] A. Sud, M. Foskey, and D. Manocha. Homotopy-preserving medial axis simplification. *Int. J. Computational Geometry & Applications*, 17:423–451, 2007.

[SS04] A. Scharf and A. Shamir. Feature-sensitive 3d shape matching. In *Proc. Computer Graphics International (CGI'04)*, pages 1530–1552, 2004.

[STA] Stanford 3d scanning repository. `http://graphics.stanford.edu/data/3Dscanrep/`.

[Stu09] Astrid Sturm. *Geometric Approximations in Two- and Three Dimensional Space*. PhD thesis, Institute of Computer Science, Free University Berlin, 2009.

[Tro] Trolltech AS. Q Public License (QPL). `http://doc.trolltech.com/3.0/license.html`.

[VM04] Gokul Varadhan and Dinesh Manocha. Accurate minkowski sum approximation of polyhedral models. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 392–401, Washington, DC, USA, 2004. IEEE Computer Society.

[Wei09] Ron Wein. 2d Minkowski sums. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009.

[Yvi09] Mariette Yvinec. 2D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.5 edition, 2009.