Doctoral Thesis

3D Voronoi skeletons
a semi-continuous implementation of the 'symmetric Axis transform' in 3D euclidean space

Author(s):
Näf, Markus

Publication Date:
1996

Permanent Link:
https://doi.org/10.3929/ethz-a-001790398

Rights / License:
In Copyright - Non-Commercial Use Permitted
3D Voronoi Skeletons
A semi-continuous Implementation of the 'Symmetric Axis Transform'
in 3D Euclidean Space

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of
Doctor of Technical Sciences

presented by
MARKUS NÄF
Dipl. Informatik-Ing. ETH,
ETH - Swiss Federal Institute of Technology
born 12th of December, 1965
citizen of St. Gallen-Straubenzell and Erlen(TG)

accepted on the recommendation of
Prof. Dr. Olaf Kübler, examiner
Prof. Dr. Leila De Floriani, co-examiner
Dr. Gabor Székely, co-examiner

November 1996
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>1</td>
</tr>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Zusammenfassung</td>
<td>4</td>
</tr>
<tr>
<td>1 The Symmetry Axis Transform</td>
<td>5</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>1.2 Shape representation</td>
<td>7</td>
</tr>
<tr>
<td>1.3 The Symmetry Axis Transform</td>
<td>8</td>
</tr>
<tr>
<td>1.3.1 Definition</td>
<td>8</td>
</tr>
<tr>
<td>1.3.2 The Skeleton represents local object symmetries</td>
<td>9</td>
</tr>
<tr>
<td>1.3.3 The Skeleton represents the topological structure of the object</td>
<td>12</td>
</tr>
<tr>
<td>1.3.4 Skeletal shape taxonomy</td>
<td>14</td>
</tr>
<tr>
<td>1.3.5 Stability and Uniqueness</td>
<td>15</td>
</tr>
<tr>
<td>2 Skeletonization Techniques</td>
<td>17</td>
</tr>
<tr>
<td>2.1 Skeleton Extraction from Distance maps</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Thinning</td>
<td>21</td>
</tr>
<tr>
<td>2.3 Analytical fire front propagation</td>
<td>22</td>
</tr>
<tr>
<td>2.4 Analytical computation for special object classes</td>
<td>22</td>
</tr>
<tr>
<td>2.5 Voronoi skeletons</td>
<td>25</td>
</tr>
<tr>
<td>3 3D Voronoi Skeletons</td>
<td>28</td>
</tr>
<tr>
<td>3.1 Object representation</td>
<td>28</td>
</tr>
<tr>
<td>3.1.1 Discrete representation of continuous shapes</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2 Cell complexes</td>
<td>31</td>
</tr>
<tr>
<td>3.1.3 Extraction of boundary points</td>
<td>36</td>
</tr>
<tr>
<td>3.2 Voronoi diagrams and Delaunay triangulations</td>
<td>37</td>
</tr>
<tr>
<td>3.2.1 Definition of Voronoi diagrams and Delaunay triangulations</td>
<td>39</td>
</tr>
<tr>
<td>3.2.2 Fundamental properties of Voronoi diagrams and Delaunay triangulations</td>
<td>40</td>
</tr>
<tr>
<td>3.2.3 The combinatorial complexity of Voronoi diagrams and Delaunay triangulations</td>
<td>41</td>
</tr>
<tr>
<td>3.2.4 Algorithms constructing Voronoi diagrams and Delaunay triangulations</td>
<td>41</td>
</tr>
<tr>
<td>3.3 Computation of the Delaunay Triangulation</td>
<td>49</td>
</tr>
<tr>
<td>3.4 Computation of the Voronoi Diagram</td>
<td>63</td>
</tr>
<tr>
<td>4 Regularization</td>
<td>64</td>
</tr>
<tr>
<td>4.1 The regularization in 2D</td>
<td>66</td>
</tr>
<tr>
<td>4.2 Homotopic equivalence in 3D</td>
<td>68</td>
</tr>
<tr>
<td>4.2.1 Deletability criterion for Delaunay polyhedra</td>
<td>69</td>
</tr>
<tr>
<td>4.2.2 Deletability criterion for Voronoi faces</td>
<td>74</td>
</tr>
<tr>
<td>4.3 The processing order</td>
<td>76</td>
</tr>
</tbody>
</table>
CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Importance measures in 3D</td>
<td>81</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Local Importance measures</td>
<td>81</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Global importance measures</td>
<td>86</td>
</tr>
<tr>
<td>4.5</td>
<td>The dimensionality of skeletal constituents</td>
<td>97</td>
</tr>
<tr>
<td>5</td>
<td>Postprocessing</td>
<td>99</td>
</tr>
<tr>
<td>5.1</td>
<td>From Voronoi faces to skeletal sheets</td>
<td>99</td>
</tr>
<tr>
<td>5.2</td>
<td>Regeneration of the shape</td>
<td>102</td>
</tr>
<tr>
<td>5.3</td>
<td>The hierarchical skeleton</td>
<td>104</td>
</tr>
<tr>
<td>5.4</td>
<td>Illustration of the postprocessing on a synthetic object</td>
<td>107</td>
</tr>
<tr>
<td>6</td>
<td>Results and applications</td>
<td>111</td>
</tr>
<tr>
<td>6.1</td>
<td>Skeletons of artificially created shapes</td>
<td>111</td>
</tr>
<tr>
<td>6.2</td>
<td>Bone thickness characterization using skeletonization</td>
<td>113</td>
</tr>
<tr>
<td>6.3</td>
<td>Analysis of the cortical brain structure</td>
<td>115</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>127</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary</td>
<td>127</td>
</tr>
<tr>
<td>7.2</td>
<td>Outlook</td>
<td>129</td>
</tr>
<tr>
<td>A</td>
<td>Voronoi diagram</td>
<td>132</td>
</tr>
<tr>
<td>A.1</td>
<td>Proofs of fundamental Voronoi diagram properties</td>
<td>132</td>
</tr>
<tr>
<td>B</td>
<td>Boundary sampling</td>
<td>135</td>
</tr>
<tr>
<td>B.1</td>
<td>Sufficient sampling density for r-regular shapes</td>
<td>135</td>
</tr>
<tr>
<td>B.2</td>
<td>Sufficiency of the boundary sampling</td>
<td>141</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>143</td>
</tr>
<tr>
<td>Curriculum Vitae</td>
<td></td>
<td>154</td>
</tr>
</tbody>
</table>
Abstract

In Computer Vision shape takes a fundamental place. It can be regarded to be the result of the low-level image processing stages and at the same time it forms the basis for high-level vision primarily concentrating on scene analysis and interpretation. This thesis investigates the description of highly complex biological shape in 3D Euclidean space. The method goes back to a suggestion of Blum who argued that natural shape is the result of growth processes which can be captured by local object symmetries. The resulting shape description method is well known in computer vision under the terms skeleton, stick figure, thinned shape, medial axis transform or symmetric axis transform (SAT) – the term used in this thesis. Blum gave an intuitive definition of the SAT in the continuum with the prairie-fire analogy: The object to be analyzed is modeled by a patch of grass and its boundary is set on fire. The skeleton is the locus where the firefronts meet and quench. This definition can be naturally extended to any dimension. The challenge lies in the implementation of the concept, which is defined in the continuum, on the discrete image raster. While in 2D many implementations of the SAT have been proposed, their generalization to 3D has still not been solved satisfactorily. This thesis presents the generalization of the 2D Voronoi skeletons – a specific implementation of the SAT – to 3D and describes the power and the weakness of the SAT concept in 3D. With our method it is for the first time possible to generate reasonable 3D skeletons of highly complex biological shape, e.g. of the white brain matter.

Voronoi skeletons exploit the intimate relation between the Voronoi diagram – one of the most fundamental data structures in computational geometry – of the boundary points of an object and its skeleton. The generation of Voronoi skeletons includes three steps: extraction of a sufficiently dense set of boundary points, computation of the Voronoi diagram of these points and extraction of the skeleton as a subgraph of the Voronoi diagram. Since the Voronoi diagram takes a fundamental place in the presented implementation of the SAT an overview of the state of the art in the computation of Voronoi diagrams is given. One of the key problems addressed in this thesis is a technical one: the lack of efficient procedures to generate the Voronoi diagram of several 100'000 points. The presented algorithm is an extension of standard procedures of computational geometry which is not only able to deal with such an amount of data but also with the fact that most of the points are in degenerate position. The extraction of the skeleton from the Voronoi diagram – commonly called regularization – and the problems thereby encountered is another essential part of this thesis. The regularization iteratively reduces the Voronoi diagram to the Voronoi skeleton by removing 'unimportant' parts. The regularization needs a mean to guarantee at every moment that the reduced Voronoi diagram is homotopically equivalent to the initial one. We have developed such homotopy checks based on local information of the Voronoi diagram. Additionally the regularization relies on importance measures expressing the significance of skeletal parts. We distinguish between local and global importance measures. While local measures basically make explicit the 'amount' of local object symmetry a skeletal part represents, global ones additionally take into account what part of the overall shape depends on that skeletal part. In this context we identify the reason why it has been relatively easy to define global measures in 2D and no way has been found up to now to generalize them to 3D. It lies in the special topological structure of the 2-dimensional Voronoi diagram. As a consequence the hierarchic organization of 3D shape cannot be made explicit by automatic procedures similar as in 2D.

Finally we propose a postprocessing of the resulting Voronoi skeletons to compile the many microscopic Voronoi faces into larger unbranched skeletal sheets and to establish a shape hierarchy by manual interaction. The developed tools are assessed on synthetic objects, as well as on some very complex biomedical shapes extracted from MR data.
Zusammenfassung


Schliesslich wird eine Nachbearbeitung der Voronoi Skelette beschrieben, welche einerseits die vielen, oft mikroskopisch kleinen Voronoi Polygone zu grösseren unverzweigten Skelettflächen zusammenfasst und es andererseits erlaubt, interaktiv eine Formhierarchie zu erstellen. Die entwickelten Werkzeuge wurden getestet anhand künstlich erzeugter Objekte und anhand einiger höchst komplexer biologischer Formen, welche aus MR Daten extrahiert wurden.
Chapter 1

The Symmetry Axis Transform

1.1 Introduction

In Computer Vision shape takes a fundamental place. It can be regarded to be the result of the low-level image processing stages and at the same time it forms the basis for high-level vision primarily concentrating on scene analysis and interpretation. That's why good shape description is crucial in developing 'intelligent' systems and much research effort has been spent on it. Unfortunately shape and more precisely human perception of shape is perhaps one of the less well-understood phenomena. This is reflected by the many existing theories of shape description and recognition, each attempting to explain some specific aspect of the problem. As a consequence there exist many shape representation techniques in computer vision and, given the complexity of the task, no single general method has emerged to address all aspects of the problem. Hence in practice the appropriate shape description method must be chosen from case to case.

The choice of a specific shape description method should be guided by

- The type of shape to be described

  On one side we have man-made objects. Their boundary is well approximated by polygons. Resulting from manufacturing processes, they are often composed of simpler basic elements and the topological structure of their surface can become quite complex.

  On the other side we have biological objects. They usually show a smooth boundary which can hardly be approximated by polygons. As being influenced by the process of growth they can become quite complex and folded while retaining a relatively simple topological structure.

- The task to be solved by the shape processing

  Two major problem tasks have been identified to be relevant in shape processing. One of them is shape matching in which the amount of equivalence of two shapes has to be decided. For this purpose the emphasis has to be put on global shape features. The other one is shape analysis and characterization in which local features must be used to identify the individual and characteristic variation of shape. Somewhere in the middle of these two lies shape classification or naming. In this case global features can help to make a rough distinction between different shape classes but local features must be used to distinguish members of the same class.

Classical computer vision generally deals only with 2D images and tries to reconstruct 3D scenes from 2D views. However the need for 3D tools appears as soon as object representation is concerned. Besides that in medical image analysis, which is our field of application, sensors have been developed that are able to deliver truly 3-dimensional data. That's why we perceive a general need to extend image processing methods from 2D to 3D. At the same time the importance of shape description methods is even higher in medical image analysis than in other
fields since biological shape is often so complex that a priori knowledge in the form of anatomical models is needed already at the low-level image processing stages. So in this thesis we are concentrating on the description of complex biological shape resulting from 3D radiological data. The task we have in mind are both shape matching and shape characterization.

In his seminal work [20] Blum argues that classical projective and coordinate geometries are not satisfactory as 'a natural geometry for biological shape'. He claims that the development of biological shape is characterized by growth processes and proposes a sort of growth geometry to account for this evolutionary point of view. The basic property captured by this growth geometry is local object symmetry and the resulting shape description method is known in computer vision under the terms skeleton, medial-axis transform, symmetric-axis transform, stick figure or thinned shape. What makes skeletal shape representation so attractive in 2D is its potential to unify the above mentioned complementary requirements to capture global shape features on one side and local ones on the other side by means of the hierarchical organization of skeletal parts. This allows for a task-dependent differentiation between global and local shape features. While in 2D there are fast and robust skeletonization procedures available the same cannot be said for the 3D case although we expect skeletons to have a major impact in 3D than in 2D. In fact there exist very powerful boundary-based methods in 2D which cannot be directly generalized to 3D due to the more complex theory of differential geometry in higher dimensions. We will see that the basic concept of local object symmetry is easily generalized to higher dimensions. Fundamental problems arise in the implementation and specifically in the definition of shape hierarchy. We will give a detailed analysis of these problems in chapter 4. Despite all problems we will demonstrate on such complex objects as the human brain that we have been able to turn 3D skeletonization from a fascinating idea into a valuable tool of volumetric image analysis.

A central question that must be addressed when a method defined in the continuum has to be used by computers is the the definition of discrete shape and the approximation of the continuous theory in discrete space. This is especially crucial in shape description which should be able to faithfully represent the Euclidean geometry of objects represented in a discrete image raster. The benefit in using Euclidean geometry is rotational invariance of the outcoming shape descriptor. Note that the vast majority of skeletonization methods, e.g. thinning, that have been generalized to 3D are operating on the discrete image raster and therefore do not achieve rotational invariance of the produced skeletons. On the other hand some attempts have been made to implement skeletonization analytically on objects represented in the continuum (e.g. in computer aided design). The resulting methods, however, are highly complex and only applicable to man-made objects which can be represented by a few polygonal patches. What we propose is to steer a middle course and adopt a semi-continuous approach. The discrete objects are approximated by a discrete set of boundary points but then all computations on this point set are performed in the continuum. There are two important advantages in contrast to discrete techniques as thinning. First the resulting skeleton is invariant under geometric transformations. Secondly we obtain a graph-like representation of the skeleton as an intrinsic by-product of the method. Hence there is no need to transform a heap of unorganized voxels into a structure suited to graph traversal routines and semantic interpretation.

At least a word of advice has to be given at this point. Skeletonization, from the conceptual viewpoint, has two serious drawbacks. First it is defined on binary objects, i.e. segmentation is a prerequisite for using skeletonization. Secondly, and this makes the first drawback even more critical, it is very sensitive to topological noise and dependent from the mutual position of objects in a complex scene, i.e. objects embedded one in another or touching one another. In chapter 6 we will point out the possibility to combine skeletal representation with the complete set of local object symmetries. The pros and cons of the two approaches are complementary and we think that a symbiosis of the two could overcome the drawbacks of both of them. Hence our interest into skeletons has to be seen with respect to the combined application of both representations. This thesis covers the skeletonization part as the first step on the way there and for a detailed discussion of the other parts one should look into [174].
1.2 Shape representation

The representation of shape can be approached basically from two different ways by which any object can be unambiguously described:

- using the boundary it is surrounded by and
- using the region it occupies.

It is clear, that – at least for objects with non self-intersecting surfaces – these two descriptors are equivalent in the sense, that any of them can be uniquely generated if the other one is known.

In the past the shape description methods that have been developed can be classified, according to these two views of the object's shape, into boundary-based and region-based methods. The most popular boundary-based methods include the following:

- Objects can be represented by parameterized continuous models as e.g. superquadrics [139, 181] or Fourier-series expansions [140, 98]. Methods have been developed to make these representations invariant under affine transformations.

- The boundary of an object can be described by the union of surface patches. Many different breeds of splines has been developed [61] and used successfully for curve and surface modeling. There are two notable special cases of spline representation which play an important role in shape description, representation of objects by straight line segments (i.e. polygons in 2D) and by piecewise constant curvature spline segments (i.e. combination of lines and circles in 2D).

- The mathematically well understood tools of differential geometry can be used. An example are the codon shape descriptors used by Hoffmann [76, 77] who defines contour segments between curvature minima. Another work to be mentioned in this context is the curvature primal sketch by Asada and Brady [7, 8], based on the localization and analysis of curvature extrema along the boundary.

On the other hand there is also a large class of region-based methods available:

- Constructive Solid Geometry (CSG) representation [168], where objects are coded as set-theoretical combination of simple solids. This method is widely used for CAD representation of man-made objects but has clear limitations when applied to natural shape. Similar representation can be constructed for the description of object boundaries [148].

- Ribbons [127, 154] in 2D as well as generalized cylinders [19] in 3D are very popular methods for object coding in image analysis. As they are based on the idea of sweeping a geometric figure (described parametrically) along a spine or axis (coded e.g. as a 1D spline), they are especially suited for the description of axial symmetry.

- Momentum descriptors (cf. [81, 180]) that represent an object by a set of its central moments. These moments can be made invariant to affine transformations like the Fourier descriptors.

- Skeletons representing local object symmetries. The skeletonization techniques will be overviewed in the next chapter.

Traditionally, these two aspects of the object's shape are analyzed separately with very different mathematical means as listed above. However the theoretical equivalence of these descriptors has been demonstrated by Leyton [106] who showed that (internal and external) skeletal branches are emerging from curvature extrema of the object's boundary. This basic theorem establishes a duality relation between the differential geometric characterization of the boundary and the branches of its skeleton.

Procedures for boundary description are usually simple, efficient and based on the well established mathematical theory of the differential geometry of curves. Consequently, they are much more widely used for object characterization in different applications. There are, however, some deficiencies of boundary-based descriptors which limit their usefulness in practical applications:
1.3 The Symmetry Axis Transform

1.3.1 Definition

The Medial Axis Transform (MAT) or Symmetry Axis Transform (SAT) has been proposed by Blum [20, 21, 22] in the late sixties for the generation of shape descriptors capturing axes of axial growth. We will introduce it directly in d-dimensional space while giving some illustrations only in 2D for the sake of simplicity. The shape descriptors generated by the SAT are generally called skeletons. Since an object can be characterized equivalently by the area inside and outside of its boundary two types of skeleton can be defined for any selected object boundary, an *inside skeleton* and an *outside skeleton* as illustrated on Figure 1.1. During this work the expression *skeleton* will always denote the inside skeleton unless stated otherwise. The SAT can be defined in two alternative ways for continuous objects:

---

**Figure 1.1:** A binary object (left) and its inside and outside skeleton together with the object’s boundary (right).

- differential geometrical analysis is very sensitive to noise;
- occlusion may seriously disturb boundary-based descriptors;
- they are not appropriate to catch global shape features (as e.g. symmetry) and to make them explicit;
- they can very hardly reveal the hierarchical organization of the shape.

The concept of skeletonization should be able to help exactly at the points listed above. However, due to other limitations, it certainly cannot replace boundary-based shape descriptors, but complement and support them. As the dual concepts of the object’s boundary and region carry information in a redundant way, they look at the same target from a different side. Specific questions can be answered more easily by one or by the other concept, and one should always use the most elegant and convenient way. The ultimate goal of shape analysis should be the usage of boundary and region based information in a redundant way for optimal object characterization and recognition.
1.3 The Symmetry Axis Transform

Figure 1.2: Skeleton of a rectangle as the centers of maximal balls. The boundary of the object is shown in bold lines, its skeleton is drawn by thin lines. Two maximal balls are illustrated, their centers are denoted by empty, the touching points between the balls and the object border by filled dots. An included non-maximal ball, which is shown as a dotted circle.

- A very illustrative definition of the SAT can be given using the prairie-fire analogy. The object to be analyzed is modeled by a patch of grass and its boundary is set on fire. The skeleton is the collection of locations where the generated fire fronts meet and quench each another.

- The skeleton is the location of the centers of maximal hyperspheres (in the following called balls) placed within the object. For the analysis only those balls are taken into account which are completely included by the object to be skeletonized. Let

\[ B(x_0, r) \subseteq \mathbb{R}^d, \quad B(x_0, r) = \{ x \in \mathbb{R}^d | \delta(x, x_0) \leq r \} \]

be a ball with center \( x_0 \) and radius \( r \) using the (usually Euclidean) distance measure \( \delta(x, y) \). For the object \( O \subseteq \mathbb{R}^d \) the set of included balls is

\[ B = \{ B(x_0, r) | B(x_0, r) \cap O = B(x_0, r) \} \]

a ball will be called maximal if it is not completely covered by any other ball in this set:

\[ \mathcal{M} = \{ B \in B \mid \forall B_i \in B, B_i \neq B \rightarrow B_i \cap B \neq B \} \]

The skeleton can then be defined as:

\[ S = \{ x_0 | B(x_0, r) \in \mathcal{M} \} \]

Figure 1.2 illustrates the skeleton definition by the centers of maximal balls on an example in 2D. The boundary of the object (a rectangle) is shown in bold lines, its skeleton is drawn by thin lines. Two maximal balls are illustrated, their centers are denoted by empty, the touching points between the balls and the object border by filled dots. An included ball, which is, however, not maximal, is shown as a dotted circle.

While the prairie-fire analogon is very intuitive, for most mathematical operations the definition through the maximal balls is more suitable. The equivalence of the two has been proved by Calabi and Hartnett [35, 37, 73, 36].

1.3.2 The Skeleton represents local object symmetries

The skeleton formation process can be interpreted in a way, that the maximal balls express the local symmetry between the points at which they touch the object boundary. This local symmetry property is projected onto the center of the ball since it is the locus to which the symmetric points have equal distance.

In 2D there are basically two different ways in which a maximal disk touches the boundary of the object (cf. left image of figure 1.3):
1.3 The Symmetry Axis Transform

Figure 1.3: The left image shows the skeleton of a rectangle with a rounded side. Two maximal disks are shown, one touching the boundary at two distinct points and therefore representing mirror symmetry, the other touching the boundary at a circular arc (bold arc) and hence representing rotational symmetry. The right image shows the skeleton of a cylinder having a spherical end (only half of the cylinder is shown in order to make the skeleton visible). Three maximal balls are drawn. The first one touches the cylinder only at two points and therefore is lying on a skeletal sheet which represents local mirror symmetry. The second one touches the cylinder along the bold drawn circle and lies on a skeletal line representing axial symmetry. The third one intersects the object at a half sphere and hence represents rotational symmetry.

1. The disk touches the boundary at two distinct points. In this configuration the center of the disk makes part of a skeletal line and it represents mirror symmetry between the two boundary points.

2. The disk touches the boundary at a circular arc. Then the center of the disk is an end point of a skeletal line and represents rotational symmetry between the boundary points on this circular arc. If the object is itself a disk the skeleton degenerates to a single point which expresses the rotational symmetry of a disk.

There is of course the case that the disk touches the boundary at more than two distinct points. This is a degenerate case where neither mirror nor rotational symmetry is present.

In 3D we have to distinguish three ways in which a maximal sphere touches the boundary of the object (cf. right image of figure 1.3):

1. The sphere touches the boundary at two distinct points. In this configuration the center of the sphere makes part of a skeletal sheet and it represents mirror symmetry between the two boundary points.

2. The sphere touches the boundary at a great circle. Then the center of the sphere lies on a skeletal line and represents axial symmetry between the boundary points of this circle.

3. The intersection of the sphere with the object boundary is a connected patch of the sphere's surface. Then the center of the sphere is an end point of a skeletal line or sheet and
represents rotational symmetry of the boundary patch. If the object is itself a sphere the skeleton degenerates to a single point which expresses the rotational symmetry of a sphere.

In 3D, too, there are degenerate cases which cannot be given one of these symmetry types. A complete classification of the skeletal constituents of 3D skeletons, based on the characterization of the maximal balls belonging to the constituents, can be found in [29, 165, 166].

One has to be aware of the fact that the SAT is actually a symmetry selection process, i.e. not all local object symmetries will be represented in the skeleton but only a (small) part of them.

Large parts of the general symmetry information about the object are suppressed by the procedure. This can be easily seen in figure 1.4 for the case of a rectangle. The full set of symmetry axes is shown. The vertical line in the middle represents the symmetry axis of the two smaller rectangle sides. Its formation is blocked in the SAT by the 'quicker' burning up between the two longer sides. This feature is essential for the skeletal description and is mainly responsible for the branching topography which is the basis of the skeleton hierarchy. On the other hand, changes in the object which deeply influence the burning pattern will result in basic reorganization of the skeletal branches. The notorious sensitivity of skeletons to topological noise is a well known consequence of this, and we expect this situation to be even worse in three dimensions. The mutual position of objects (e.g. objects embedded one in another or touching one another) in a complex scene will lead to similar changes in the object's skeleton. This problem can only be avoided if the objects are segmented and isolated before skeletonization. This requirement may make skeletons nearly useless for the recognition of objects in complex scenes.

On the other hand the full symmetry set can be computed. Calculation of symmetry transforms of images using mixed wave/diffusion processes [158] or the Multi-scale Medial Axis (MMA) transformation [123] have been presented and combined with the SAT they could offer an appealing way to cope with many of the inherent problems of skeletonization. The basic idea of these general symmetry transformations is the propagation of edge information into the object, creating local symmetry or medialness information where evidence generated by opposite object boundaries accumulates without terminating front propagation at quenching points. These methods can process directly gray-valued images and also allow the boundary points to participate in any number of symmetry relations. This way one gets the complete symmetry set without the suppression effect observed in the SAT.

Figure 1.5 shows the 3D medialness field of a binary rectangle, as generated by the MMA transformation. The single images illustrate the medialness information on different scales. The images demonstrate that both symmetry axes between the long and the short sides survive, providing complete information about the symmetry of the original object.

The different symmetry axes can be found as 1D ridges in the 3D medialness space (note, that the medialness field has an extra dimension in addition to the spatial one, the scale). Figure 1.6 shows the result of ridge extraction in the 3D medialness field and a projection to the plane of the spatial coordinates. One can clearly see the classical skeleton as part of the symmetry structure, while the remaining symmetries have been constructed, too. Note that while the full symmetry set can overcome the topological sensitivity of the SAT it has the problem that the number of local symmetries is exploding. In chapter 6 we will see that one should aim at a combination of
1.3 The Symmetry Axis Transform

Figure 1.5: Medialness measure of a rectangle on different scales.

Figure 1.6: Ridges in the 3D medialness field of a rectangle. In the right image the detected ridges have been projected to the subspace of the spatial coordinates.

1.3.3 The Skeleton represents the topological structure of the object

If we now investigate the generated skeleton on a somewhat more complicated example as the one shown on Figure 1.7, one has to realize, that different parts of the skeletal branches resulting from the SAT can be grouped into two categories:

- Several parts of the local symmetry axes (they are shown as continuous lines) express real symmetry between opposite parts of the object’s boundary. As we move along the skeletal axes, the corresponding touching points between the balls and the boundary are moving along with the same amount as the center of the ball. Such skeletal parts will be called symmetry branches.

- On other parts of the symmetry axis (denoted by dotted lines) the touching points between the boundary and the maximal balls are always the same. These branches actually repre-
1.3 The Symmetry Axis Transform

Figure 1.7: Skeleton of a rectangle with a protrusion. The object’s boundary is denoted by bold line. Parts of the skeleton representing real mirror symmetry are shown as continuous lines while the “connection branches” are dotted lines. The corresponding maximal balls are also shown.

Figure 1.8: The skeleton of a key. The bold cycle corresponds to the topological skeleton of the key.

sent symmetry only between two corresponding boundary points. Hence there is “little” symmetry information encoded in them. Such skeletal parts just ensure the preservation of topology during the SAT and play a decisive role in the hierarchical organization of the symmetry axes. Accordingly we will call these skeletal parts connection branches.

The separation of symmetry and connection branches is usually not as straightforward as in this simple example. In the absence of directional discontinuity along the boundary no such clear-cut distinction can be made between skeletal parts. Still, if observing the speed of the movement of touching points between the maximal balls and the boundary while sweeping the center of the circles along the axes, we will find patches where these movements will be approximately equally fast, while on other parts the movement along the boundary will be much slower. This phenomenon allows a fuzzy distinction between the above two categories of skeletal branches.

Thanks to the connection branches the skeleton has the very important property of homotopical equivalence with the original shape. It has been shown, that this property can be proved by techniques of mathematical morphology for shapes with closed skeletons [38, 112]. While the class of shapes that can be represented by closed skeletons never had been exactly characterized, it can be shown that the requirement of having a boundary consisting of simple differentiable curves having at most a locally finite number of critical points is sufficient.
That the topological structure of the object is represented in the skeleton can be seen very well in figure 1.8. The skeleton of this key contains a cycle (dark outlined) which represents the hole of the key. All branches of the skeleton except those making up this cycle can be deleted without changing the topological structure of the skeleton. The fact that the skeletal branches of the cycle cannot be deleted corresponds to the fact that the boundary of the key can only be represented by two unconnected contours. We shall call the part of the skeleton which cannot be further reduced without topological changes the topological skeleton. It should be noted, that objects with the topology of a circle (i.e. consisting of one single component without a hole) have no uniquely defined topological skeleton. Any point of their skeleton can be regarded as the topological skeleton of the object. In the presence of many holes in the object the topological skeleton has a somewhat more complex structure as illustrated on Figure 1.9. The three holes in the key result in a condensed cyclic system as the topological skeleton on the right example while the three cycles are separated and connected by still undeletable skeletal branches on the right side.

Figure 1.9: The topological skeleton of topologically more complex objects.

In 3D the situation is a bit more complicated. The uniqueness of the topological skeleton is not only lost for objects with spherical topology, where similarly to the 2D case any point on the skeleton can be regarded as the topological skeleton, but also for objects having handles, where any closed line on the medial surface can represent the object’s topology equally well, as illustrated by figure 1.10.

During methodological and algorithmic developments in the past, homotopical equivalence between the object and its skeleton was regarded absolutely central and essential. It should be noted, however, that the importance of topological equivalence has probably been overestimated. Especially if one thinks about the fact that there are skeletal parts of the skeleton (the connection branches) which do not represent real object symmetries but are just here to reflect the topological structure of the object. In 2D one can argue that they play a decisive rule in the automatic generation of a skeletal hierarchy. In 3D however a skeletal hierarchy cannot be automatically generated anyway as we will see in this thesis. Hence in future research the requirement of homotopical equivalence between the object and its skeleton should perhaps be handled in accordance with the requirements of a specific application and no more taken as an a priori condition.

1.3.4 Skeletal shape taxonomy

The skeletal constituents produced by the SAT can then be used to decompose the object into a collection of sub-objects each drawn from a small set of primitives. Blum [21, 23] has given such
1.3 The Symmetry Axis Transform

1.3.5 Stability and Uniqueness

Two major requirements that good shape descriptors have to fulfill are uniqueness and stability. Uniqueness means that for each shape one and only one shape descriptor exists. The stability relates changes in the shape to changes in the shape descriptor. Good stability means that small changes in the shape cause only small changes in the shape descriptor. Hence similar shapes should have similar shape descriptors.

Due to the maximal disk definition, the uniqueness of skeletal representation is simple to prove. However, due to the following main sources of instability, skeletons behave extremely unstable.

- Skeletal representation is extremely sensitive to topological changes, which can be caused by segmentation noise or by the presence of other objects in the scene embedded in the one under investigation. As mentioned before, this is a fundamental problem of skeletal representation and could possibly be resolved by the combination of the skeleton with the full symmetry set of the object.

- Small geometrical disturbances also lead to dramatic changes in the skeletal structure by
generating new skeletal branches. Figure 1.11 demonstrates this problem on an example showing the contour of a rectangle is modulated by a periodical boundary disturbance. The skeletal part representing the rectangle is clearly visible and denoted by bold lines. As the frequency of the disturbance increases (and it can be made arbitrarily high), the branches caused by the modulation will be more and more dense. As the branches will be present at any positive amplitude (we deal with a continuous case), the perturbation of the object can be made arbitrarily small while (arbitrarily many) spurious branches in the skeleton remain stable. This problem can only be resolved by detecting spurious branches in the skeleton and removing them while preserving the topological structure of the object. To decide whether a specific skeletal branch is spurious or not one needs an importance measure. In the literature many different measures have been proposed. We call this process of stabilizing the skeleton regularization and will describe it in detail in chapter 4. The regularization makes skeletonization a semi-continuous mapping.

As a consequence, we cannot speak about stability without assigning an importance measure to the individual skeletal branches. This, however, raises a problem, as the importance measure can be (more or less) arbitrarily selected. This means, that different measures may lead to different skeletons, which questions the uniqueness, unless one agrees on the relevance measure.

In the example of figure 1.11 skeletal branches can be grouped into two classes. The skeletal parts shown as bold lines correspond to the major symmetries of the modulated rectangle and are identical to the skeleton of an undisturbed rectangle. Hence they should be left unchanged by the regularization process. However, from the point of view of object representation the spurious branches denoted as thin lines may even be regarded superior, as they allow a more precise object reconstruction than the stable branches. An importance measure could be defined which favors these branches.

This example shows, that additional constraints on skeleton formation are necessary, if we expect reproducibility from the representation. Such criteria are based on the topological structure of the skeleton and will be provided in the chapter on regularization.
Chapter 2

Skeletonization Techniques

The SAT is defined as a continuous transformation in the continuous space. Hence in order to make the method work on objects originating from discrete images a discrete analogue to the SAT has to be found. Much research efforts have been devoted to the development of robust and accurate skeletonization techniques. The basic procedures that have emerged can be classified into four groups:

- The definition based on the maximal inscribable disks calls for distance based methods. Formal mathematical theory has been developed [32, 34, 66, 33, 122] describing the SAT as the projection of singularities of the generalized distance function, which can also be used to generate the full symmetry set. While direct implementations of the generalized distance transform has already been tried by Wright [185], usual techniques are based on the generation and study of discrete distance maps and we will limit our overview to them.

- The prairie-fire analogy calls for physically based methods modeling fire front propagation. We distinguish between the methods which analytically determine the front propagation and the loci of quenching fire fronts and the methods based on some digital simulation of the front propagation. Thinning procedures are the simplest and most prominent and widely used representants of this class of algorithms.

- For special classes of objects the skeleton can be computed analytically in the continuum. This approach is especially suited to man-made objects as appearing e.g. in solid modeling.

- Finally a close relation between Voronoi diagrams and skeletons has been noted in 2D by Kirkpatrick [91] and by Boissonat [25] in a paper on the problem of estimating a polygonal shape from a set of points. The skeleton can be extracted from the Voronoi diagram of a set of boundary points or boundary segments representing the object. The Voronoi diagram is defined in the continuous space and hence after the approximation of the object by its boundary points the discrete image raster can be abandoned. The approach is therefore semi-continuous.

Skeleton generation algorithms have to take care of the following two aspects of the symmetry axis generation:

- the geometrical one, requiring that the symmetry axes are in their geometrically correct positions, i.e. in the “middle” of the object;

- the topological one, forcing the skeleton to retain the topology of the original object.

The topological aspect has its theoretical background in the theory of digital topology based on neighborhood relations defined on the digital image raster between adjacent pixels or voxels.

In order to speak about geometry, we have to define a metric on the image raster. Regular metrics (based on the above mentioned neighborhood relations) are very simple, and easy to use

\footnote{for a thorough definition see section 3.2}
for discrete images. These distance measures approximate the continuous $L_1$ and $L_\infty$ metrics, which are not invariant to the rotation of the coordinate system. Consequently, skeletons generated on the basis of regular metrics lack rotational invariance, which is highly undesirable. The use of Euclidean distance leads to implementational difficulties, but results in skeletons invariant to rotation and translation of the object. Since this is a fundamental requirement one poses for shape descriptors we will only consider techniques which use Euclidean or nearly Euclidean metrics.

2.1 Skeleton Extraction from Distance maps

As mentioned in the introduction to this chapter, invariance requirements make the use of Euclidean ($L_2$) metric for the generation of distance maps unavoidable. However, it is difficult to reconcile Euclidean metric with raster-type connectivity. That’s why some special metrics on the basis of weighted distance transforms using a discrete set of possible movements generated by a local mask have been proposed to approximate the Euclidean metric. A prominent example is the Chamfer metric [119, 26]. The Chamfer metric has also been generalized to 3D [27] and is widely used although algorithms producing exact or at least nearly exact distance transforms are available.

The generation of Euclidean distance maps on raster images goes back to Danielsson [45]. He proposed an algorithm which assigns to each pixel a vector pointing to one of the nearest pixels on the boundary of the object. The pixels on the object boundary are first initialized with a null vector. In a second step the vector labels are propagated to their neighbor pixels according to a sequential scanning strategy. When a smaller vector label arrives at a pixel, the vector label of this pixel is replaced with it. Danielsson proposed a scanning strategy which produces nearly error-free distance maps. Since Danielsson many other label propagation techniques have been proposed [144, 134, 186, 124]. Some of them are able to generate error-free distance maps.

Several algorithms have been proposed for the generation of 3D Euclidean distance maps on the discrete image raster [143, 124, 144]. While error-free Euclidean distance maps can be generated, they are very expensive in terms of memory consumption and computational complexity. Faster algorithms, generalizing the sequential scanning strategy of Danielsson to 3D have also been proposed [144, 41], but no rigorous analysis of the maximal error bounds has been performed.

The procedure for skeleton generation based on distance maps can now be summarized as follows:

1. The boundary of the object is extracted and represented by discrete pixels/voxels on the raster, having neighbouring pixels/voxels belonging to the object or to its background. A null vector is assigned to the boundary pixels/voxels.

2. An Euclidean distance map of the object is generated on the complete image raster using one of the above mentioned scanning algorithms.

3. In the last step pixels/voxels representing centers of maximal balls have to be extracted. When using regular metrics in 2D, such pixels correspond to the local maxima of the discrete distance map [183] which are easy to extract. Unfortunately, this is not the case for an Euclidean distance map, where more sophisticated methods have to be applied for skeleton extraction.

Figure 2.1 shows the Euclidean distance map of the boundary of an airplane. It can be well seen (especially on the surface-like representation of the function graph on the right), how the skeleton corresponds to the ridges on the distance surface [117, 177].

The detection of these ridges in the distance map can be made through several methods:

- the generalization of local extrema for height surfaces (graphs of the distance function);
- the definition and analysis of principal directions along associated lines of curvature;
- the analysis of the level sets of functions;
2.1 Skeleton Extraction from Distance maps

Figure 2.1: The discrete Euclidean distance map of the 2D outline of an airplane. On the left, the map is shown as a gray-valued image (light areas correspond to large distances from the boundary), on the right the distance function is presented as a shaded surface of the function graph.

- the analysis of the first and second fundamental form leading to nonmetric ridge definitions.

All these definitions rely on the differential geometric analysis of the distance map as a continuous function \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) in \( d \) dimensions. Hence standard techniques from differential geometry can be used. One possibility is given by the analysis of the Hessian matrix \( H(f) \) of the function \( f \). The Hessian matrix is defined as follows:

\[
H(f) = 
\begin{pmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\
\frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2}
\end{pmatrix}
\]

The definition of ridges can then be given using the eigenvectors \( (e_1, e_2, \ldots, e_d) \) and eigenvalues \( (\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d) \) of the negative Hessian matrix \(-H(f)\). A point \( x_0 \) is an \( m \)-dimensional ridge point of the function \( f \) if \( f \) has a local maximum at the point \( x_0 \) in every direction within the subspace of the \( m \) largest eigenvectors of the negative Hessian matrix. On an ideal ridge, the function \( f \) changes its value only in directions orthogonal to the ridge while it remains constant in directions along the ridge. In this case the rank of the Hessian matrix is exactly \( m \) leading to \( \lambda_i = 0 \) for all \( i > m \).

The practical implementation of this idea needs a discrete version of the proposed procedure. On a digital distance map one cannot expect to find ideal ridges. Accordingly, ridges can be identified in 2D by searching for points \( x_{i,j} \) on the image raster, where the Hessian matrix has one large negative and one small (approximately zero) eigenvalue. The eigenvector corresponding to the large eigenvalue will define the orthogonal direction to the ridge. The digital approximation of the Hessian matrix can be determined by calculating finite differences after a Gaussian smoothing of the distance map with an appropriately selected \( \sigma \).

Several attempts have been made for skeleton extraction based on distance maps [45, 50, 5, 2, 27, 6]. Still, no really satisfactory solution has been found due to two fundamental problems of the approach:

- The ridge extraction is a non-trivial task, especially for the less prominent ridges in the distance map. Only heuristic solutions (in principle "beautifying procedures") have been found for differentiation between real ridges corresponding to important skeletal branches and the ones caused by undesired effects of boundary discretization and approximations of the Euclidean distance [156].
• Distance ridge extraction has no natural built-in mechanism to guarantee topology preservation. Actually, in most cases the applied procedures result in disconnected skeletal parts.

In 3D the situation is even worse since one has to search for 2D ridges (sheets). All the procedures from 2D which are based on ridge tracking are very hard to generalize and implement as one must search for surfaces with unknown parametrization.

![Figure 2.2: Searching for quenching points in distance maps. The procedure is illustrated in 2D but can be carried out in any dimensions. Hypothetical boundary voxels are denoted by dashed areas. The point O is checked for being a ridge point by searching for the vectors OP between O and its nearest boundary voxel P and that of the neighbours within a selected environment (NQ, e.g.). The voxel O is characterized by measuring the angle φ between the vectors and by the distance d between the boundary points.](image)

As a solution to this problem the search for quenching points in the digital distance map has been proposed. From the generation of the Euclidean distance map by propagating vector labels referring to boundary points the information about the nearest boundary points is always at hand as a by-product of these algorithms and can be used for searching areas in the distance map where influence zones between different boundary points meet. The idea actually equals the generation of a "digital Voronoi Diagram" and was first proposed for regular metrics in 2D [4]. It has been shown to be expandable to quasi-Euclidean 3D distance maps [175]. Similar ideas have been used for skeletal point extraction from n-D distance maps [108] by analyzing the corresponding boundary points as delivered by the label propagation in a region around the voxels analyzed. On a distance ridge, one expects contributions from very different boundary elements on nearby skeleton points. Accordingly, one can check the vectors to the nearest boundary elements at neighbouring voxels of the distance map. The probability, that a voxel corresponds to a ridge on the digital distance map can be expressed by two criteria as illustrated by Figure 2.2:

• the angle φ between the vectors;
• the Euclidean distance d between the boundary points

We will see in chapter 4 that these criteria correspond to the basic ideas of local measure definition. After the combination of these criteria to an overall measure of significance, thresholding is used for the extraction of skeletal pixels. Of course, due to the use of a local significance measure, no topology preservation can be expected. Topological errors are tried to be kept minimal by hysteresis thresholding techniques.
2.2 Thinning

Theoretically, exact Euclidean distance transform techniques based on multi-label propagation [124] would be optimal to extract voxels cutting the influence region of more than one boundary points providing more precise information about the presence of a possible ridge. The use of such techniques for distance ridge extraction is, however, limited by the computational expense of the necessary calculations.

As a summary we can say that distance map based methods produce geometrically accurate skeletons while their connectedness is lost. A possible solution to this problem with the help of elastically deformable models has been proposed in 2D by Leymarie and Levine [103, 104, 105]. The original object contour is regarded as a snake [86] which is then deforming itself while searching for ridges in the object’s distance map. The topological correctness can be ensured during the deformation process while the correct geometry is guaranteed by the adjustment onto the distance ridges. The problem here is the same as for thinning procedures discussed below: one must detect the endpoints of the skeleton where the deforming snake has to stop.

2.2 Thinning

The classical way of simulating front propagation is to use thinning procedures. Thinning is basically a topology-preserving object reduction technique. Two basic approaches exist which can provide the theoretical background for the development of thinning procedures:

- mathematical morphology [162, 111] based on the erosion of binary figures by a structuring element which is usually a ball;
- the theory of digital topology based on connectivity on the digital image raster which has first been developed for 2D [136, 150, 151, 153, 95, 97, 74] and then extended to 3D [120, 121, 155].

While mathematical morphology provides the theoretical basis for the study of the skeletonization process and the properties of skeletons, the most widely used methods for thinning in the skeletonization literature belong to the second class [178, 47]. A lot of sequential [3, 49, 138] and parallel [152, 137, 47, 75] thinning algorithms have been published in the literature and used for skeleton generation in 2D. Similarly several 3D thinning procedures have been proposed and implemented, based on local deletability criteria [182, 52, 17, 18, 67].

The theory of connectivity concentrates basically on object topology, without geometrical considerations (actually the topological relation is purely built up on the basis of pixel/voxel neighbourhoods and can be done without any use of distances). The preservation of object topology can be checked locally during pixel/voxel deletion by a discrete set of deletion rules. These rules have to be extended in order to prevent the deletion of pixels from skeletal lines once the thinning has reached one-pixel wide lineal structures in 2D and additionally to prevent the deletion of voxels from the boundary of skeletal sheets in 3D [107]. This line-end and sheet-boundary preservation actually corresponds to the detection of fire-front quenching.

The sequence of the pixel deletion in a thinning procedure is only partially given by the request for topology preservation. Additional constraints on the deletion sequence trying to imitate a peeling-like process can help to keep the skeleton at least approximately in the middle of the object leading to geometrically more acceptable results [52, 16].

Thinning is a relatively fast and algorithmically simple way to generate topologically correct skeletons. While thinning techniques have been used for the generation of 3D skeletons for generic object shapes [149], their performance is proved to be practically satisfactory only for the analysis of strongly elongated structures as blood vessels [176, 94] or thin (sheet- or line-like) structures as the gray-matter envelop of the cerebral cortex [110, 109].

The applicability of thinning procedures for generic objects is limited by the following problems:

- Even complex heuristics for selecting an optimal deletion sequence were not able to achieve geometrically correct skeletonization in all cases.
• The differentiation between important skeletal features and noise branches is even more
difficult as in ridge-based skeletonization on distance maps. The usage of heuristic "beau¬
tifying rules" [156] can only partially solve this problem.

Summarizing one can say that thinning procedures are able to fulfill the requirement for
topologically correct skeletons while their geometric accuracy is unsatisfactory. There have been
some attempts to combine the distance map based methods with thinning procedures. One
possibility is to additionally constraint thinning procedures. In this case the pixels representing
the centers of maximal balls are determined in a separate preprocessing step and are declared non-
deletable in the following thinning procedure [1]. This way, the thinning procedure takes care of
the homotopical equivalence producing connection paths between the fragmented distance ridges.
The distance map can even provide further support for an optimal approximation of peeling while
selecting the deletion sequence in the pruning [183, 177]. Such algorithms however imply a lot of
heuristics and do not provide a coherent framework to solve the problems of distance map and
thinning based skeletonization. Most notably, we need a theoretical basis for the formulation of
pruning procedures, i.e. for the characterization of the importance of different skeletal branches.

### 2.3 Analytical fire front propagation

The propagation of fire fronts can be computed analytically. This means solving a partial dif¬
ferential equation (the diffusion equation), with the contour as initial value. The SAT is the
limiting solution of the equation, i.e. in a second step one has to detect singularities caused by
the quenching of fire fronts which is similar to the ridge detection in distance maps. Considerable
effort has been made to create stable numerical schemes for the solution of the equation using
finite difference techniques [133, 163, 40, 58]. Different approaches have been published in the
literature to apply these techniques for skeleton generation in 2D, as e.g.

* Object shape has been characterized through the physical analogy to conservation laws
  and shock formation [88, 89, 90]. Lately, stable numerical schemes have been developed
  providing a basis for the creation of computer algorithms for singularity detection during
  front evolution [72, 169, 170].

* Classical variational approaches (as the finite element method) used in analyzing elastic-
  plastic torsion problems lead also to methods of calculating ridges (i.e. skeletons of simply
  connected domains bounded by disjoint arcs [65].

* Recent studies in orientation-based representations for mathematical morphology have also
delivered first results for the description of dilatation fronts [51].

In spite of the first promising results in 2D cited above, these techniques have rarely been used
in practical implementations of the SAT in 2D and even less in 3D. Most of them suffer from the
same problems as the ridge detection on distance maps, i.e. the outcoming skeletons are often
disconnected.

### 2.4 Analytical computation for special object classes

Already among the first algorithms presented in the literature to compute the SAT in 2D there
have been two algorithms which directly compute the continuous skeleton for objects represented
by polygonal chains. One algorithm is due to Kirkpatrick [91], the other to Lee and Drysdale [101,
100]. These algorithms have been carried on by Srinivasan and Nackman [171] to be used for
multiply connected polygonal chains with holes and for planar straight line graphs by Mayya
and Rajan [114]. All these algorithms have been inspired by the observation that the skeleton of
a polygon can be seen as a generalized Voronoi Diagram which is a well known structure from
Computational Geometry. The Voronoi diagram of a set of points in the plane is a partition of the
plane so that each region of the partition is the locus of points which are closer to one point from
the set than to any other. In this definition we have italicized the notion point since this is the part that will be generalized in order to get a generalized Voronoi Diagram which coincides with the skeleton. Hence the basic idea is to use a generalized Voronoi diagram generation procedure which takes not only points but also line segments as input sites. The nature of the Voronoi edges which separate two regions of the generalized Voronoi diagram is then determined by the nature of the two input sites of these regions. A (point,point) or a (line,line) pair generates a Voronoi edge which is a straight line, namely the perpendicular bisector of the two sites. A (point,line) pair generates in general a parabolic arc unless the point is an endpoint of the line or lies in the prolongation of the line segment. The generation procedure of the generalized Voronoi diagram (=skeleton) of a polygonal chain can then be summarized as follows:

- an initial list of generalized Voronoi edges is generated by considering the three sites (line, point, line) associated to a pair of adjacent line segments of the polygonal outline. For each generalized Voronoi edge the starting point, the initial direction and the generating sites are maintained.

- The edge list holds the generalized Voronoi edges which are only partly determined. Hence in a second step these edges are completed by tracing lines or parabolic curves up to the next Voronoi vertex. If this vertex is new all edges coming out of it are determined and filled into the edge list. This step is repeated for each edge in the list until all generalized Voronoi edges are determined.

An extension of this algorithm to compute the SAT of a multiply connected planar region bounded by line segments and circular arcs is due to Gursoy and Patrikalakis [71, 135].

Due to the computational complexity involved in 3D only recently and only a few algorithms have been proposed. One was developed by Hoffmann [79], who proposed a method for assembling the skeleton of a CSG (Constructive Solid Geometry) object. His method proceeds by determining points of closest approach between pairs of boundary elements and checking these points to make sure they are in fact on the skeleton (each of these points are on the skeleton if and only if the distance to the pair of elements is less than or equal to the distances to the other boundary elements). The points are then sorted in order of increasing distance from the boundary, and then a local analysis around each point is performed, in attempt to identify whether the point lies on a face, edge or vertex of the skeleton. This determination is made by identifying all boundary elements which lie at the same minimum distance from the point and forming a set of simultaneous equations in n variables which describe the equidistantial set that the point belongs to. Based on the rank of the Jacobian of this set of equations, the point is predicted to lie on a face, edge or vertex of the skeleton. Then neighbouring faces and edges are traced out in order of increasing distance from the boundary. As one can see this method is very tedious to implement and computationally expensive. Furthermore it produces only an approximation of the skeleton.

Another method to approximate the skeleton of polyhedra in 3D has been proposed by Bertin and Chassery [15]. They first study theoretically the type of bisecting surfaces that can be generated in polyhedral objects between polygon, line and point sites. It turns out, that only quadric surfaces have to be expected: paraboloid of revolution, hyperbolic paraboloid, parabolic surface, cone and plane. Then they give a practical implementation which approximates the exact skeleton of such objects by discretizing each boundary polygon of the objects with a set of points, then computing the Voronoi diagram of the point sets and suppressing the parts of the Voronoi diagram which have been produced by points lying on the same polygonal patch.

Reddy and Turkiyyah [147, 146] also propose an algorithm to compute the skeleton of a 3D polyhedron. Their method is an attempt to generalize the generation procedures for generalized Voronoi diagrams in 2D. They compute an abstract Delaunay triangulation of the polyhedron and use the result to obtain by duality the generalized Voronoi diagram. The algorithm can explicitly determine certain critical points of the skeleton, but does not contain accurate representations of the curves and surfaces making up the skeleton.

As far as we know there has been only one publication by Sherbrooke, Patrikalakis and Brisson [167] describing an algorithm to compute the exact skeleton of 3D polyhedra. The algorithm
2.4 Analytical computation for special object classes

uses a classification of the different types of skeletal points which arise in the case of polyhedra in order to generate an adjacency graph of faces, edges and vertices which represent the skeleton of a polyhedral solid. The classification scheme is an extension to the one introduced by Brandt [29]. The different points are called junction point, seam point, sheet point, seam-end point and end points (those lying on a rim). This classification is illustrated on the simple skeleton of a box (see figure 2.3). Each skeletal point has a set of governors associated. These are the boundary elements (the faces, the non-convex edges and the vertices of non-convex edges of the polyhedron) from which the skeletal point is equidistant. The skeleton is then constructed by determining one vertex after the other together with the curves (called seams) joining the vertices. This is a recursive process that starts at one of the polyhedron's vertices belonging to an edge which is convex and therefore makes part of the skeleton. Then the seam starting at this vertex can be traced out using the offset sweep equations of the seam's governors. The offset sweep of a boundary element describes the geometric location of its equidistant points for any fixed distance \( r \). The offset sweep of a point is a growing sphere, for an edge it is a growing cylinder and so on. The seam is traced until a junction point is reached which in turn can be used to start over tracing the outgoing seams. Once the vertices and seams have been traced, the sheets can be determined by a simple traversal of the structure. The sheets will be trimmed quadric surfaces, where the trimming curves are the bounding seams and convex edges.

The algorithm produces the exact skeleton. Its major limitations are that the polyhedral object is not allowed to have handles and the whole procedure has been used only for objects having less than a hundred boundary elements. Hence it seems to be limited to problems arising in solid modeling and can hardly be applied to biological objects.

Finally Attali [10] proposes a new class of continuous shapes which she calls polyballs. A polyball is a finite union of balls. The exact skeleton of polyballs can be computed from the centers of some balls, the connection lines between the centers of some pairs of balls and the Voronoi diagram of a set of singular points on the polyball's surface. However the question remains what type of objects can be meaningfully represented by polyballs.
2.5 Voronoi skeletons

As we have seen up to now the fundamental problem that must be solved when transferring the SAT from the continuum to the discrete image raster is to reconcile the raster-type connectivity with the Euclidean metric. The skeletons produced by many of the classical skeletonization methods suffer either from the lack of connectivity or from geometrical inaccuracy. When the concept of the Voronoi skeleton emerged [93, 92, 30, 128] it offered a promising way to skeletonize digital images in a geometrically correct way while retaining the topological structure of the original objects. The concept is due to an intimate relation between the skeleton of an object and the Voronoi diagram of a point set representing the boundary of the same object.

Figure 2.4: Full Voronoi diagram generated by the points resulting from a low resolution sampling of a box surface. The infinite Voronoi faces have been cut.

The Voronoi Diagram\(^2\) of a discrete point set in the d-dimensional space is a partition of the space into cells so that each cell of the partition contains exactly one member of the point set and is the locus of all points which are nearer to this member than to any other. In this way the space is divided into convex cells which are not always finite. In 3D the Voronoi cells are called Voronoi polyhedra and the member of the point set which is inside the Voronoi cell is called generating point. The faces which divide two Voronoi polyhedra are called Voronoi faces. A Voronoi face has equal distance to the generating points of the two Voronoi polyhedra it bounds. The edges of a Voronoi face are called Voronoi edges. They have equal distance to all generating points of the Voronoi polyhedra they belong to. Finally the vertices of the Voronoi polyhedra are called Voronoi vertices. Again they are equidistant from all generating points of the Voronoi polyhedra they belong to. Figure 2.4 illustrates the Voronoi diagram of a point set in 3D.

The Voronoi skeleton concept can easily be understood by a prairie-fire analogy, similar to the continuous one. The only difference is that not the continuous outline of the object, but a discrete number of boundary points are set on fire. If the fire is evolving isotropically, circular fire-fronts are generated at each of these boundary points. These fire-fronts will exactly quench at the boundaries of the point set’s Voronoi cells. In other words the Voronoi diagram of a point set \(P\) equals the exact skeleton of the object which is defined by the complement of the point set. Formally

\[
VD(P) = Sk(R^d \setminus P)
\]

Boissonat and Kofakis [25] noted that the skeleton of an object \(O\) which is described by a set of boundary points \(P \subseteq \partial O\) can be approximated from a subgraph of the point set’s Voronoi

\(^2\)for a thorough definition see section 3.2
diagram. Brandt and Algazi [30] studied the quality of this approximation in terms of the sampling density. Schmitt [157] has shown by morphological methods that if the density of boundary points uniformly goes to infinity the corresponding Voronoi diagram converges to the object's exact skeleton. Hence the following relation can be stated

\[ \text{Sk}(\mathbb{R}^d \setminus \partial O) = \lim_{P \to \partial O} VD(P) \]

Figure 2.5: The three major steps of Voronoi skeleton generation illustrated on an example in 2D (upper row) and in 3D (lower row). The figures a show the approximation of the object with its boundary points, the figures b the generation of the Voronoi diagram (the full one in 2D and the one restricted to the inner of the object in 3D) and the figures c the skeleton extracted from the Voronoi diagram during the regularization.

On the basis of these results the generation of Voronoi skeletons can be described by the following three major steps:

1. Approximation of the object's boundary by a sufficiently dense set of points. These points are called the generating points. The extraction of the generating points is the only step performed on the discrete image raster. The following steps are done in the continuous space. Hence the generation of Voronoi skeletons can be seen as a semi-continuous process.

2. Generation of the Voronoi diagram of these generating points. It is sufficient to compute the Voronoi diagram inside the object if we are only interested in the 'inner' skeleton. The 'inner' Voronoi diagram can be determined in different ways:
   - The Voronoi diagram can be intersected with the object.
   - The Voronoi vertices lying inside the object and the other elements of the Voronoi diagram attached to theses vertices can be taken.
   - The Voronoi elements which correspond to dual Delaunay elements lying inside the object can be chosen.

We use the second of these methods.
3. Extracting the skeleton from the Voronoi diagram as a subgraph. We will call this process regularization. This is the crucial and most difficult part of the whole procedure. That's why we will dedicate a separate chapter of this thesis to it.

The method is illustrated on a 2D and on a 3D example in figure 2.5.

Up to now there have been only a few attempts to implement Voronoi skeletons in 3D. The first is due to Yu, Goldak and Dong [187]. They simply used the Voronoi vertices as an approximation to the Voronoi skeleton and did not use any regularization to get rid of the discretization effects and to distinguish spurious skeletal sheets from important ones. Brandt [29] went one step further and introduced a simple regularization scheme based on the radius function of skeletal sheets. He states that spurious skeletal sheets have always a rapidly changing radius function. The removal of such skeletal parts is done without any guarantee to preserve the homotopy between the original object and the resulting skeleton.

The closest work to ours is the one of Attali [11, 10, 9]. The main differences are

- we don't use subsampling of the object boundary but extract the boundary points at the full resolution of the image raster.

- we are able to deal with cospherical points in the computation of the Delaunay triangulation. Note that cospherical points cannot be avoided since the generating points are lying on a regular grid (in our case the image raster). The solution adopted by Attali to this problem is to add noise to the generating points. However the noise will also affect the outcome skeleton.

- for the regularization we defined rules by which the homotopical equivalence between object and skeleton can be guaranteed when removing arbitrary convex Delaunay polyhedra and not only Delaunay tetrahedra (this is essential when cospherical generating points are allowed).

- we propose another importance measure for the regularization

- we developed a postprocessing which compiles the still numerous Voronoi faces into large unbranched skeletal sheets and dispose of tools which support the user to establish a hierarchy between these sheets that reflects the shape's hierarchy.
Chapter 3

3D Voronoi Skeletons

3.1 Object representation

3.1.1 Discrete representation of continuous shapes

The simplest way to acquire a digital image of a real-life scene is sampling. An example device which uses sampling during image acquisition is an MR scanner which measures magnetic properties of small volumes of the investigated objects. The most common method of sampling uses a regularly spaced set of points of measurement. The position of these sample points is usually given by a rectangular grid. Another sampling geometry that sometimes has been used in 2D takes the sampling points from a hexagonal grid. In our case we will always deal with samples coming from an isotropic rectangular sampling grid. Such a set of samples can then be represented in computers as an array of numbers. The sample values are quantized to a set of gray levels; thus a digital image can be regarded as an array of integers or floats (according to the degree of quantization), called the digital image raster.

The elements of the digital image raster are called picture elements or pixels in 2D and volume elements or voxels in 3D. Using an isotropic rectangular sampling grid, the pixels are squares and the voxels cubes of length \( h \) which is called the pixel (voxel) spacing. A coordinate system is defined so that each pixel (voxel) midpoint receives according to its position \((i, j, k)\) in the digital image raster an integer coordinate \((h \cdot i, h \cdot j, h \cdot k)\) defining its position in Euclidean space.

Once a digital image of a real-life scene has been acquired, the object to be described by its skeleton must first be extracted from the image. This process is called segmentation. It consists in marking the pixels (voxels) which belong to the object. The marking process can be thought of as creating a mask or an overlay, congruent with the digital image, in which there are marks at positions corresponding to the object points. We can represent this overlay as a binary image of the same size as the original digital image with 1's at object points and 0's at background points. Segmentation is one of the largest research fields in digital image processing and there is still no standard approach to it. Several methods have been proposed reaching from manual over user-guided to fully automatic segmentation. Let us assume for our work that the object of interest has already been segmented by any of these methods in a preprocessing step.

In order to compute the Voronoi skeleton of the object, now represented as a binary image, we have to approximate its boundary by a set of boundary points. This raises two questions:

- how might the boundary of a binary object be defined?
- what quality of approximation should be used to approximate the object's boundary? This question will be addressed later in this section.

To answer the first question topological notions as adjacency and connectivity have to be defined on the digital image raster. Going back to a suggestion of Rosenfeld [150] adjacency between pixels (voxels) is usually defined by the following different kinds of neighborhoods. In 2D

- the four horizontal and vertical neighbors at distance \( h \) are called 4-neighbors.
3.1 Object representation

- the four diagonal neighbors at distance $\sqrt{2}h$ together with the previous 4-neighbors are called 8-neighbors.

and in 3D

- the six neighbors in the coordinate directions at distance $h$ are called the 6-neighbors
- the twelve neighbors with a common voxel edge and being at distance $\sqrt{2}h$ together with the 6-neighbors are called the 18-neighbors.
- the eight neighbors with a common voxel vertex and being at distance $\sqrt{3}h$ together with the 18-neighbors are called 26-neighbors.

With these neighborhood definitions the so-called neighborhood graphs can be introduced. The nodes of a k-neighborhood graph ($k = 4, 8$ in 2D and $k = 6, 18, 26$ in 3D) are the pixel (voxel) midpoints and the edges are defined by the midpoints of any two pixels (voxels) being k-neighbors of each other. The connectivity of a set of pixels (voxels) is then defined by the connectivity of the corresponding neighborhood graph. If the set of pixels (voxels) is described by a connected k-neighborhood graph, it is called k-connected. Hence to speak about connectivity in the digital raster using neighborhood graphs one always has to agree upon which type of neighborhood graph is used for object pixels (voxels) and which one is used for background pixels (voxels). Rosenfeld [151] has shown that different neighborhood graphs must be used for the object and its background in order that fundamental topological theorems as the Jordan curve theorem 1 can be transferred to digital images. Nevertheless a problem arises in the definition of the boundary between the object and its background. The boundary of an object is defined by those elements which are adjacent both to the object itself as also to its background. In 2D this leads already to four(!) different boundary definitions:

- an inside boundary if the boundary pixels are required to belong to the object and an outside boundary if they belong to the background.
- the boundary will be 4-connected if 8-adjacency has been used to define the boundary pixels and 8-connected otherwise.

The four different boundaries for a simple 2D object are illustrated in figure 3.1.

![Figure 3.1: Four different boundary definitions for the outlined object. (a) inside boundary with pixels being 8-adjacent both to the object and to its boundary (b) inside boundary with pixels being 4-adjacent both to the object and to its boundary (c) outside boundary with pixels being 8-adjacent both to the object and to its boundary (d) outside boundary with pixels being 4-adjacent both to the object and to its boundary](image)

1The Jordan curve theorem states that a simple closed curve in the Euclidean plane cuts the plane into two separated connected components.
3.1 Object representation

Figure 3.2: The Voronoi diagram of a simple object’s outside boundary (on the left) and of it’s inside boundary (on the right) are different.

Figure 3.3: The inner Voronoi diagram of an object can become disconnected due to the boundary definition by pixels

Note that the different types of neighborhood graphs used for the object and for its neighborhood not only lead to multiple possibilities of boundary definitions but also causes the boundary to be sensitive to the notion of foreground and background. This has its effects on the Voronoi diagram which will be different for the points of the object’s inside boundary and for the points of the background’s inside boundary as can be seen in figure 3.2. Hence it doesn’t help to chose just one of the four definitions. A second problem is that the so-defined boundaries have the same dimension as the object to which they belong while according to our intuition they should be 1-dimensional for 2D objects and 2-dimensional for 3D objects. As a consequence the inner Voronoi diagram of an object’s inside boundary points can become disconnected inside the object (see figure 3.3). In 3D these problems are similar. A simple solution is to represent the digital image raster and consequently the objects defined on it not only by pixels (voxels) but as a structure consisting of elements of different dimensions: 0-dimensional vertices, 1-dimensional edges, 2-dimensional squares and 3-dimensional cubes. A voxel e.g. consists of one 3-dimensional cube, six 2-dimensional squares, twelve 1-dimensional edges and eight 0-dimensional vertices. We can
then use the 2-dimensional squares which are common to both a voxel of the object and a voxel of the background to describe the boundary of a digital object in 3D. The so-defined boundary is invariant to the notion of object and background and is 2-dimensional as it should be. Moreover this definition has a mathematically sound foundation in the concept of cell complexes from combinatorial topology. We formally introduce the theory of cell complexes in the following subsection since it provides us not only a framework to formalize topological notions as connectivity, boundary etc. on the digital image raster but also on the Delaunay triangulation and on the Voronoi diagram.

3.1.2 Cell complexes

Cell complexes and their generalization, the CW-complexes are a basic concept of algebraic topology. In image processing they can be used to represent the basic structure of images, the raster grid, as a topological space. This is important since in image analysis often topological notions as connectedness, adjacency and boundary are needed. The representation of images as cell complexes allows to take such concepts from the classical algebraic topology which has the advantage that they are consistent. We will use cell complexes to represent not only the digital image raster and digital objects in two and three dimensions but also the Voronoi diagram and the Delaunay triangulation of digital objects. In this section we introduce some basic definitions from the theory of cell complexes following the ideas of Kovalevsky [97, 96]. Thereby we proceed in such a way that the exact mathematical definitions of abstract cell complexes are given and contemporarily illustrated on a specific cell complex, the two-dimensional digital image raster $I^2$. Finally we will also define the three-dimensional digital image raster $I^3$ as a cell complex and later in this chapter the three-dimensional Delaunay triangulation $DT$ and its dual, the three-dimensional Voronoi diagram $VD$.

We start by giving the definition of an abstract cell complex:

**Definition 1 (abstract cell complex)**

An abstract cell complex $C(C, B, \text{dim})$ is a set $C$ of abstract elements provided with an antisymmetric, irreflexive and transitive binary relation $B \subseteq C \times C$ called the bounding relation and with a dimension function $\text{dim} : C \rightarrow \mathbb{I}$ from $C$ into the set $\mathbb{I}$ of the non-negative integers such that $\text{dim}(c') < \text{dim}(c'')$ for all pairs $(c', c'') \in B$.

The bounding relation $B$ is a partial order in $C$. It indicates the ordered pairs $(c', c'')$ of elements such that $c'$ is said to bound $c''$, which is denoted by $c' \prec c''$. A cell is said to bound a cell of higher dimension if it is part of its boundary, i.e. if it is a face of it. The function $\text{dim}$ defines the dimension of each element. An element $c'$ with $\text{dim}(c') = d$ is called a $d$-dimensional element or a $d$-cell. A complex is called $k$-dimensional or a $k$-complex if the dimension of all its elements is less or equal to $k$.

Let us illustrate this on the cell complex $I^2$. $I^2$ consists of vertices, edges (the line segments joining two vertices, often called 'raster cracks') and faces (usually called 'pixels'). They make up the set $C$. The dimension function assigns 0 to each vertex, 1 to each edge and 2 to each face. The bounding relation is chosen in such a way that a vertex bounds an edge, which in turn bounds a face. For example in the left image in figure 3.4 which shows a small part of $I^2$ pixel $f_0$ is bounded by the four raster cracks $e_0, e_1, e_2, e_3$ and the four vertices $v_0, v_1, v_2, v_3$. $e_0$ is bounded by the two vertices $v_0$ and $v_1$, $e_1$ by $v_0$ and $v_2$ and so on.

Along with the definition of an abstract cell complex we introduce the term 'subcomplex'. A subcomplex $C'(C, B', \text{dim}')$ of a given complex $C(C, B, \text{dim})$ is a complex whose set $C'$ is a subset of $C$ and the relation $B'$ is the intersection of $B$ with $C' \times C'$. The dimension function remains unchanged for all $c' \in C'$ : $\text{dim}'(c') = \text{dim}(c')$. This means that to define a subcomplex $C'$ of $C(C, B, \text{dim})$ it suffices to define the corresponding subset $C'$ of the elements. Thus it is possible to speak of a subcomplex $C' \subset C$ while understanding the subcomplex $C'(C', B', \text{dim}')$. This is important because it allows us to regard the subcomplexes of $C$ as subsets of $C$ and thus it is possible to use the common formulae of the set theory to define intersections, unions and complements of subcomplexes of one and the same complex $C$. 
On the cell complex $I^2$ a digital object $O$ can be defined as a subcomplex $O \subseteq I^2$. In the right image in figure 3.4 the elements $v_0, \ldots, v_7$, $e_0, \ldots, e_9$ and $f_0, \ldots, f_2$ are a subset of the whole set of vertices, edges and faces of $I^2$ and together define one specific digital object $O$.

In the following we will concentrate on finite cell complexes, i.e. complexes containing only a finite number of elements. This restriction reflects the fact that in practice raster images and objects defined in them are always of finite nature. We will show now the relation between the fundamental topological notion of a topological space and the cell complexes according to definition 1. For that purpose let us recall the definition of a topological space:

A topological space $(X, T)$ is defined as a set $X$ of abstract elements and a system $T = \{S_0, \ldots, S_i, \ldots\}$ of subsets $S_i$ of $X$. These subsets are called the open sets of the space and must satisfy 3 fundamental axioms:

[A1] $X \in T$ and $\emptyset \in T$

[A2] the union of an arbitrary number of sets in $T$ must be contained in $T$

[A3] the intersection of a finite number of sets in $T$ must be contained in $T$

A topological space is called separable (more precisely T0-separable) if it satisfies also the axiom:

[A4] For any two elements $x_1, x_2 \in X$ there exists in $T$ such an open subset $S'$ that exactly one of the elements is in $S'$.

The T0-separability guarantees that for any two space elements exists a subdivision of the space into two complementary parts so that each part contains exactly one of the two elements. Kovalevsky has proven the following theorem which illustrates the importance of cell complexes

**Theorem 1** Every finite separable topological space is a cell complex as defined in Definition 1.
3.1 Object representation

This theorem shows that every finite topological space can be expressed as a cell complex. On the other hand it can also been demonstrated that every cell complex is a topological space. For that purpose the system of all open subsets of the set of elements \( C \) of a cell complex \( C \) has to be defined. These open subsets then define the open subcomplexes of \( C \). An open subcomplex is defined in the following way.

**Definition 2 (open subcomplex)**

A subcomplex \( S \) of \( C \) is called open in \( C \) if

\[
\forall (c \in S, c' \in C) : c \prec c' \Rightarrow c' \in S
\]

Again in figure 3.4 an example of an open subcomplex of \( I^2 \) is represented by the set \( \{e_6, f_1, f_2\} \).

The open subsets defined in this way obviously fulfill the axioms of a topology. Therefore any cell complex is a topological space and all notions and results of the general topology which are meaningful for finite sets may be transferred to cell complexes. We will now apply the topological notions of connectivity, boundary and adjacency to the cell complexes.

**Figure 3.5:** The topological notion of path-connectivity defines the connectivity of a cell complex. The cell complex \( C_0 \) is not connected since there exists no path between an element from subcomplex \( S_0 \) and an element from subcomplex \( S_e \).

The connectivity of a cell complex can be defined by the topological notion of path-connectivity:

**Definition 3 (connectivity)**

A sequence of elements \((c_0, \ldots, c_i, \ldots, c_n)\) of a subset \( S \) of a cell complex \( C \) is called a path in \( S \), denoted by \( c_0 \Rightarrow c_n \), if for every two elements which are adjacent in the sequence one of them bounds the other.

A subcomplex \( S \) is called connected if

\[
\forall (c_i \in S, c_j \in S) : \exists (c_{k_0}, \ldots, c_{k_m} \in S) \text{ with } (c_i, c_{k_0}, \ldots, c_{k_m}, c_j) \text{ is a path in } S
\]

For a practical implementation to determine whether an object is connected or not one uses the fact that the union of two subcomplexes \( S_1, S_2 \) which are both connected is again connected if
in each of them an element can be found so that there exists a path in \( S_1 \cup S_2 \) between them. As an example consider the two complexes \( C_1 \) and \( C_2 \) in figure 3.5. Both of them contain the two subcomplexes \( S_1 = \{ e_0, e_1, e_2, e_3, f_0 \} \) and \( S_2 = \{ e_4, e_5, e_6, e_7, f_1 \} \), but \( C_2 \) additionally contains the vertex \( v_0 \). \( S_1 \) and \( S_2 \) each are connected. However \( C_1 \) which is the union of \( S_1 \) and \( S_2 \) is not connected since a path in \( C_1 \) from \( f_0 \) to \( f_1 \) can only go through \( v_0 \) (white circle). For the same reason \( C_2 \) is connected.

![Figure 3.6: Illustration of the boundary definition on \( \mathcal{I}^2 \)](image)

The boundary of an object can be defined by the topological notion of neighborhood:

**Definition 4 (boundary)**

*every open subcomplex \( S \subset C \) is called a neighborhood of an element \( c \) if \( c \in S \). We write \( S = N(c) \).*

*the intersection of all neighborhoods of an element \( c \) is called the smallest neighborhood of \( c \) and written \( N_0(c) \).*

*the boundary \( \partial(S) \) of a subcomplex \( S \subset C \) relative to \( C \) is defined as follows:*

\[
\partial(S) = \{ c \in C | \exists (c' \in S, c'' \in C \setminus S) : c' \in N_0(c) \land c'' \in N_0(c) \}
\]

Figure 3.6 shows on the left the smallest neighborhoods of the 0-, 1- and 2-dimensional elements \( c^0, c^1 \) and \( c^2 \) of \( \mathcal{I}^2 \). On the right a simple object in \( \mathcal{I}^2 \) is sketched. The object faces are grey shaded, its boundary consists of the bold edges and vertices. As an example two boundary elements \( v \) and \( e \) are emphasized. The element \( e \) has in its smallest neighborhood the element \( f_0 \) which belongs to the object and the element \( f_1 \) which belongs to the objects complement. Likewise the element \( v \) has in its smallest neighborhood among others the elements \( f_2 \) and \( f_3 \) of which one lies in the object and the other not.

This boundary definition has two important properties:

1. The boundary is invariant to the notion of background and foreground.
2. The boundary of a \( k \)-dimensional subcomplex \( S \) of a \( k \)-dimensional complex \( C \) contains only elements of dimension equal to or lower than \( k - 1 \). This is because in a \( k \)-dimensional complex the smallest neighborhood of a \( k \)-dimensional element consists only of the element itself. But a single element cannot belong at the same time to a subcomplex \( S \) and to its complement \( C \setminus S \).
The last definition concerns adjacency. Two subcomplexes can be said to be adjacent if they are topologically spoken incident:

**Definition 5 (adjacency)**

Two subcomplexes $S_1, S_2 \subseteq C$ are called adjacent (written $S_1 \bowtie S_2$) if two conditions hold:

(a) $S_1 \cap S_2 = \emptyset$

(b) $\exists (c_1 \in S_1, c_2 \in S_2) : (c_1 \prec c_2) \lor (c_2 \prec c_1)$

In figure 3.5 the subcomplex $S_1 \cup u_4$ is adjacent to the subcomplex $S_2$.

As we will see in the section 3.2 on Voronoi diagrams and Delaunay triangulations the two constructs are dual representations. Duality has a precise meaning in combinatorial topology. Here comes the exact definition

**Definition 6 (duality)**

Two $n$-dimensional cell complexes $C(C, B, \dim), C'(C', B', \dim')$ are called dual if there exists a one-to-one mapping $\alpha : C \mapsto C'$ with the following properties:

if $\alpha(c_1) = c'_1, \alpha(c_2) = c'_2$ then

(a) $c_1 \prec c_2 \iff c'_1 \prec c'_2$

(b) $\dim'(c'_1) = n - \dim(c_1)$ and $\dim'(c'_2) = n - \dim(c_2)$

In the Delaunay triangulation the Delaunay polyhedra are mapped to Voronoi vertices, Delaunay faces to Voronoi edges, Delaunay edges to Voronoi faces and Delaunay vertices to Voronoi polyhedra. The duality will play a key role in the section 4.2 on homotopic equivalence during regularization.

**The 3D digital image raster as a cell complex**

The three-dimensional digital image raster $\mathcal{I}^3$ of an image of size $d_0 \times d_1 \times d_2$ consists of the following basic elements:

- The 0-dimensional elements are the $(d_0 + 1) \times (d_1 + 1) \times (d_2 + 1)$ vertices of the three-dimensional integer raster with x-coordinates in $[0, d_0]$, y-coordinates in $[0, d_1]$ and z-coordinates in $[0, d_2]$.
- The 1-dimensional elements are the $3d_0d_1d_2 + 2(d_0d_1 + d_0d_2 + d_1d_2) + d_0 + d_1 + d_2$ sets of two vertices which have the exact distance 1 between them. They are called edges.
- The 2-dimensional elements are the $3d_0d_1d_2 + d_0d_1 + d_1d_2$ sets of four vertices which form a square of unit area. They are called faces.
- The 3-dimensional elements are the $d_0d_1d_2$ sets of eight vertices which form a cube of unit volume. They are called voxels.

As bounding relation we choose the set of all pairs $(e', e'')$ of distinct elements $e'$ and $e''$ for which holds $e' \cap e'' = e'$. This relation is antisymmetric because $(e' \cap e'' = e') \land (e'' \cap e' = e'') \implies e' = e''$ which has been excluded. For the same reason it is also irreflexive. Finally it is transitive because from $e_1 \cap e_2 = e_1$ and $e_2 \cap e_3 = e_2$ follows $e_1 \cap e_3 = (e_1 \cap e_2) \cap e_3 = e_1 \cap e_2 = e_1$. The dimension we have assigned to each element is related to the number of vertices which define that specific element: $\dim(e) = \log_2(|e|), |e| \in \{1, 2, 4, 8\}$. From the definition of the bounding relation we know that $e' \prec e'' \Rightarrow |e'| < |e''| \Rightarrow \log_2(|e'|) < \log_2(|e''|) \Rightarrow \dim(e') < \dim(e'')$. 


3.1 Object representation

3.1.3 Extraction of boundary points

The boundary of the object to be described is given as a set of 2-dimensional voxel faces. For the Voronoi diagram generation we need to approximate this boundary by a set of points. One possibility is to use the midpoints of the faces. This has the advantage that the Voronoi diagram will also remain connected at places where two object voxels are only connected over a single vertex. One should note that in 3D the number of generating points produced by approximating the boundary with the resolution of the digital image raster will be extremely large (in the range of several 100,000 for images of size 256 x 256 x 256). It is natural that several attempts have been made to reduce the number of boundary points. Simple subsampling does not work since it bears the risk to lose information in critical regions of the boundary. Hence intelligent methods have been developed which subsample the boundary in a non-uniform way. It is intuitively obvious that from the point of view of regeneration of the original shape the boundary should be sampled more densely in regions of high curvature and less in flat regions. The original boundary can then be regenerated by connecting the sampling points with splines. The problem has been investigated for example by Asada and Brady [7, 8].

Figure 3.7: The skeleton of a polygonal object (a triangle) on the right and the Voronoi diagram of a set of boundary points generated by an anisotropic sampling along the boundary (left).

Unfortunately this result does not carry over to the boundary approximation for the Voronoi skeleton generation. The principle reason for this is that all the results of non-uniform sampling do not only use the positional information of the sampled boundary points but also the implicit knowledge about the direction in which the boundary has to continue from these points. In the Voronoi diagram generation however all knowledge about the object boundary is lost except the positional information of the boundary points. Hence for the generation of Voronoi skeletons one should always remember the result of Schmitt [157] which says that the Voronoi diagram of an object's boundary points will converge to the skeleton if the boundary sampling goes uniformly to infinity. Hence the only way of subsampling that can be allowed is uniform subsampling by reducing the resolution of the image raster. However, this will increase the error of approximation in the skeleton and eventually lead to the loss of small scale features or even worse to changes in the topological structure of the described object. The problems of non-uniform subsampling are illustrated in figure 3.7 for the boundary of a triangle. While we take a lot of generating points around the corners of the triangle (the high curvature areas), no samples have been taken along the straight line sides. The resulting Voronoi diagram (shown left) shows considerable deviation from the expected skeleton of the continuous object on the right. The inner Voronoi diagram is even disconnected. Hence the sampling density should always correspond to the resolution of the digital image raster.

Finally, we would like to show that sampling the object uniformly at voxel level is sufficient to guarantee that the inner Voronoi diagram which approximates the object's skeleton is homotopically equivalent to the object. According to Brandt [30] this can be guaranteed if each Voronoi region is cut by the object's boundary into exactly two simply-connected parts. In his paper Brandt used the theory of regular sets to give a proof for the 2D case. This proof relies on the assumption that the continuous shape underlying the discrete representation of the object in the image raster is r-regular. A shape is called r-regular if it is morphologically open and closed with
3.2 Voronoi diagrams and Delaunay triangulations

Voronoi diagrams and Delaunay triangulations are fundamental data structures in computational geometry. Given a set $S$ of points in $d$-dimensional space their Voronoi diagram divides the space according to the nearest-neighbor rule: each point of $S$ is associated with the region of space closer to it than to any other point in $S$. The Voronoi diagram is a dual structure to the Delaunay triangulation, which connects points that are equidistant from each other.

In the context of 3D voxel representations, Voronoi diagrams and Delaunay triangulations are used to model and analyze the boundaries and interiors of objects. The Voronoi diagram of a set of points in 3D space is defined as the set of all points that are closer to a given point than to any other point in the set. The Delaunay triangulation is a triangulation of the points such that no point in $S$ is inside the circumcircle of any triangle in the triangulation.

The homotopical equivalence between the voxel representation of an object and its inner Voronoi diagram can be directly established without the detour of $r$-regular shapes. The idea is to define a neighborhood relation between the sampling points on the $(d-1)$-dimensional object boundary. Such a neighborhood is naturally given by the $(d-1)$-dimensional Voronoi diagram of the boundary points computed on the $(d-1)$-dimensional manifold defined by the object's boundary. Then the more general condition reads as follows:

\[ \text{The } d\text{-dimensional Voronoi diagram of a sampling intersected with the object's boundary must be identical to the } (d-1)\text{-dimensional Voronoi diagram of the same sampling but computed on the } (d-1)\text{-dimensional object boundary.} \]

Then the $d$-dimensional Voronoi regions are cut into two simply-connected parts since both the original Voronoi regions and their intersections with the object's boundary (which are $(d-1)$-dimensional Voronoi regions on the object's boundary) are simply connected. The homotopical equivalence of the object and its inner Voronoi diagram can then be seen from the fact that each subpart of the Voronoi regions which lies inside the object can be continuously retracted to the part of the inner Voronoi diagram bounding that Voronoi region. These regions associated with the boundary samples collectively cover the object and the retractions in each of these regions can be constructed so that they are piecewise continuous along the seams between regions. Therefore the composite retraction which takes the object into the inner Voronoi diagram is also continuous and the two sets are homotopically equivalent.

In appendix B we show that the above required condition holds for the 3-dimensional voxel representation of any object when the boundary points are defined to be the center points of the voxel faces separating the object from the background.

In appendix B we give a detailed analysis of these problems in 2D and a generalization to 3D. It turns out that to fulfill the above condition the boundary samples must be taken from a grid which is twice as fine as the image raster. In 3D, besides a quadruplicated sampling rate, an additional adaptation of the object at places where it is only connected over a voxel corner is needed. This result, however, is only of theoretical interest since the homotopical equivalence between the voxel representation of the object and its inner Voronoi diagram can be directly established without the detour of $r$-regular shapes. The idea is to find a more general condition that must be fulfilled by the Voronoi diagram of the object's boundary points so that the intersection of the Voronoi diagram with the object cuts each Voronoi region into two simply connected regions, one belonging to the interior of the object and one to its exterior. For that purpose a neighborhood relation has to be defined between the sampling points on the $(d-1)$-dimensional object boundary. Such a neighborhood is naturally given by the $(d-1)$-dimensional Voronoi diagram of the boundary points computed on the $(d-1)$-dimensional manifold defined by the object's boundary. Then the more general condition reads as follows:

The $d$-dimensional Voronoi diagram of a sampling intersected with the object's boundary must be identical to the $(d-1)$-dimensional Voronoi diagram of the same sampling but computed on the $(d-1)$-dimensional object boundary.

Then the $d$-dimensional Voronoi regions are cut into two simply-connected parts since both the original Voronoi regions and their intersections with the object's boundary (which are $(d-1)$-dimensional Voronoi regions on the object's boundary) are simply connected. The homotopical equivalence of the object and its inner Voronoi diagram can then be seen from the fact that each subpart of the Voronoi regions which lies inside the object can be continuously retracted to the part of the inner Voronoi diagram bounding that Voronoi region. These regions associated with the boundary samples collectively cover the object and the retractions in each of these regions can be constructed so that they are piecewise continuous along the seams between regions. Therefore the composite retraction which takes the object into the inner Voronoi diagram is also continuous and the two sets are homotopically equivalent.

In appendix B we show that the above required condition holds for the 3-dimensional voxel representation of any object when the boundary points are defined to be the center points of the voxel faces separating the object from the background.
3.2 Voronoi diagrams and Delaunay triangulations

space closest to it. The Voronoi diagram has been used for a wide variety of applications as a short historical overview will make apparent. One of the first researchers giving a careful definition of this geometric construct has been M.G. Voronoi in 1908. He has been motivated by the study of quadratic forms as initiated by Gauss in 1840 and Dirichlet in 1850. In honour of the pioneering work of these mathematicians the construct has been referred to as the Dirichlet tessellation or the Voronoi diagram. Most of the early work on Voronoi diagrams has been done by crystallographers including Delaunay starting in the 1920s. They studied the problem of filling space with congruent copies of a set of crystals. In crystallography Voronoi regions are usually called domains of action according to Niggli. Metallurgists on the other hand call them Wigner-Seitz zones after Wigner and Seitz who in 1933 used them to study equilibrium properties of alloys. In geography Voronoi regions have been used for the first time in 1911 by the climatologist Thiessen to improve the estimation of precipitation. Since that Voronoi regions have also been called Thiessen polygons. Later work extended the use of Thiessen polygons to urban planning, ecology, cartography and even economy. Today’s mathematical interest in Voronoi diagrams lies in the study of tilings, finite element analysis, packing and covering problems with congruent spheres and coding theory. Most important for us is the work of Blum in 1967 who suggested the use of Voronoi diagrams as a descriptor for the shape of objects in computer vision. More recently Voronoi diagrams have become a central object in computational geometry. One important problem is the nearest-site search. It can be solved efficiently by computing in a preprocessing step the Voronoi diagram of the set of sites and then for each query locating the region of the Voronoi diagram containing the query point. This problem is best known in its geometric version under the name post-office problem (determine the nearest post-office given the location of a person). Another problem is finding the closest pair among a number of points which has an obvious application in collision detection; the two closest points are most likely to collide. Another application is clustering for which one needs to find the closest site for each of the sites. Since each point on a Voronoi edge maximizes the distance to the closest sites, Voronoi diagrams can also be used in placement and motion planning problems and hence are especially important to robotics.

Closely related to the Voronoi diagram is the Delaunay triangulation. It has already been introduced by Voronoi in 1908 for points that form a lattice and was extended by Delaunay in 1934 to irregularly placed points. The Delaunay triangulation of a set S of points in d-dimensional space is the unique subdivision of space into convex cells with vertices in S so that there are no points lying inside the hypersphere of any cell. The Delaunay triangulation is dual to the Voronoi diagram in a graphtheoretical sense. K-dimensional elements of the Voronoi diagram correspond to (d-k)-dimensional elements of the Delaunay triangulation. The Delaunay triangulation in the plane has a number of interesting properties which are used in many different applications. One property is the equiangularity which says that for any two triangles the replacement of their common edge by the alternative diagonal in the quadrilateral formed by the two triangles does not increase the minimum of the six angles of the two triangles. The Delaunay triangulation thus maximizes the minimum angle over all triangulations of the given point set. Triangulations without "extreme" angles are desirable in finite element analysis, interpolation methods and computer graphics for visualization. Another important fact is that the Delaunay triangulation is a supergraph of a number of widely used graphs spanned by a set of points in the plane. Among them are the minimum spanning tree (straight-line connection of the points with minimum total edge length) with applications in transportation problems, pattern recognition and cluster analysis, the Gabriel graph (graph consisting of those edges between two points for which the open disk having that edge as its diameter is empty of other points) with applications to processing geographical data and the relative neighborhood graph (graph consisting of those edges between two points for which no other point is closer to both of them than their interpoint distance) useful in pattern recognition. Not all of these properties however do carry over to Delaunay triangulations in higher dimensions. There is e.g. no three-dimensional analog to the equiangularity property. A related property however is that the three-dimensional Delaunay triangulation minimizes the largest radius of the smallest spheres containing a tetrahedron. Note that this is not the same as the circumradius since small but
elgtrated tetrahedra may have arbitrarily large circrmradii. Hence the Delaunay triangulation is the most compact one in this sense. Another useful property is the existence of an acyclic in-front/behind relation for the triangles of a three-dimensional Delaunay triangulation defined with respect to a fixed viewpoint. This can be used in computer graphics for hidden surface removal.

Apart from the classical Voronoi diagram and its dual Delaunay triangulation there exists a wide range of generalized constructs originating from them with yet a number of important applications. However it is not possible to mention them in this context. Readers with interest in that should refer to Aurenhammer's survey of Voronoi diagrams [12].

After this short overview of the most prominent application fields of Voronoi diagrams and Delaunay triangulations we will now proceed to give an exact mathematical definition of the two constructs, enumerate some of their most important properties, have a look at their combinatorial complexity and then review the state of the art of algorithms used for their construction.

3.2.1 Definition of Voronoi diagrams and Delaunay triangulations

The VD is defined as a cell complex \( V \) in the following way. Let \( S = \{ u_i \} \) be a set of \( n \) point sites in \( d \)-dimensional Euclidean space \( E^d \) which have been taken from the boundary of an object. For \( x, y \in E^d \) the Euclidean distance between \( x \) and \( y \) is designated by \( \delta(x, y) \). For a nonempty subset \( R \subseteq S \), the Voronoi cell \( V(R) \) is the set of all points of \( E^d \) equidistant from all sites in \( R \) and closer to every site of \( R \) than to any site not in \( R \):

\[
x \in V(R) \iff (1) \forall v_i, v_j \in R : \delta(x, v_i) = \delta(x, v_j) \\
(2) \forall v_i \in R, v_j \in S \setminus R : \delta(x, v_i) < \delta(x, v_j)
\]

If \( R = \{ u_i \} \) is a set containing only one element, \( V(R) \) is the set of all points strictly closer to \( u_i \) than to any other site. This type of Voronoi cell can also be described as the intersection of a number of halfspaces and that's how the Voronoi diagram is often constructed. A halfspace is the set of points lying to one side of a hyperplane. The points of \( E^d \) which are nearer to the first of two sites \( v_i, v_j \) are contained in the halfspace defined by the hyperplane orthogonal to the segment \( u_i u_j \) formed by the two sites and containing \( u_i \). Using this fact the Voronoi cell of a singleton set \( R = \{ u_i \} \) can be obtained by intersecting all halfspaces containing \( u_i \) and defined by pairs of sites whereof one site is \( u_i \). From this construction of the Voronoi cells follows also that they are convex.

Any point of \( E^d \) lies in \( V(R') \) for some \( R' \subseteq S, R' \neq \emptyset \). \( V(R) \) may be empty, either because there is no point equidistant from all \( v_i \in R \) or because any point equidistant from all \( v_i \in R \) is also equidistant from some \( v_j \in S \setminus R \). The basic elements of \( V \) are then all nonempty Voronoi cells \( V(R) \), for \( R \) being a subset of \( S \). The bounding relation is the set of all pairs \((V(R'), V(R))\) of distinct nonempty Voronoi cells for which holds \( R \subseteq R' \). This relation is antisymmetric and irreflexive because \((R \subseteq R') \land (R' \subseteq R) \Rightarrow R = R' \) which has been excluded. It is also transitive since from \( R \subseteq R' \) and \( R' \subseteq R'' \) follows \( R \subseteq R'' \). The dimension function can be defined by the number of linearly independent points contained in the set \( R \) defining a Voronoi cell \( V(R) \):

\[
\text{dim}_{VD}(V(R)) = d + 1 - |R|^2
\]

The Delaunay triangulation \( D \) can be directly derived from the Voronoi diagram. For every \( R \subseteq S \) with a nonempty Voronoi cell \( V(R) \) a Delaunay cell \( D(R) \) is defined as the relative interior of the convex hull of the point set \( R \).

The basic elements of the Delaunay triangulation are then all the Delaunay cells \( D(R) \), where \( R \) varies over subsets of \( S \) with \( V(R) \) nonempty. The bounding relation is the set of all pairs \((D(R'), D(R))\) of distinct Delaunay cells for which holds \( R' \subseteq R \). This relation is again antisymmetric and irreflexive. The dimension function is again defined by the number of linearly independent points contained in the set \( R \) defining a Delaunay cell \( D(R) \):

\[
\text{dim}_{D}(D(R)) = |R|^2 - 1
\]

\( |R| \) designates the number of linearly independent points in \( R \). This is the rank of a matrix whose rows are made up of the coordinates of the point sites contained in \( R \).
V and D are dual cell complexes since in the definition of the Delaunay triangulation exactly one Delaunay cell has been assigned to every Voronoi cell, the bounding relation has been reversed and \( \dim_D(D(R)) + \dim_V(V(R)) = d \).

### 3.2.2 Fundamental properties of Voronoi diagrams and Delaunay triangulations

Next we shall list a number of fundamental properties of Voronoi diagrams and Delaunay triangulations which are often used in their construction. The proofs of these properties can either be found in one of the standard textbooks of computational geometry (e.g., \([142, 55]\)) or have been given in journal articles. Since the properties P2-P4 are needed for a thorough understanding of our implementation of computing the Delaunay triangulation we will explicitly quote the corresponding proofs given by Tanemura et al. \([179]\) in appendix A.1.

**P0** In a d-dimensional Delaunay triangulation every d-dimensional Delaunay cell \( D(R) \) has an empty circumsphere.

**P1** Every nearest neighbor \( s_i \) of any site \( s \in S \) defines a face of the Voronoi cell \( V(R), R = \{ s \} \).

**P2** Consider a set of triangles (or polygons in the case of cospherical sites) \( \{ s_0, s_1, s_2 \} \) where \( s_1 \) is a nearest neighbor of \( s_0 \). Among them suppose a triangle \( \{ s_0, s_1, s_2 \} \) has the minimum circumradius. Then the triangle is a Delaunay face.

**P3** Consider a set of tetrahedra (or polyhedra in the case of cospherical sites) \( \{ s_0, s_1, s_2, s_j \} \). Among them suppose a tetrahedron \( \{ s_0, s_1, s_2, s_3 \} \) has the minimum circumradius. Then the tetrahedron is a Delaunay tetrahedron.

**P4** Suppose the tetrahedron \( \{ s_0, s_1, s_2, s_3 \} \) is a Delaunay tetrahedron. Consider a set of tetrahedra \( \{ s_0, s_1, s_2, s_j \} \) where \( s_j \) is in the halfspace \( H_f^{-s_0} \) of face \( f = \{ s_0, s_1, s_2 \} \) which does not contain the site \( s_3 \). Among them suppose a tetrahedron \( \{ s_0, s_1, s_2, s_4 \} \) has the minimum signed circumradius. The signed circumradius has the same absolute value as the circumradius and its sign is negative if \( s_4 \) lies in the same halfspace as \( s_3 \). Then the tetrahedron is a Delaunay tetrahedron.

**P5** Two opposite d-cells \( \text{conv}(R) \) and \( \text{conv}(R') \) are called locally Delaunay if \( R \) and \( R' \) both have circumspheres \( C \) and \( C' \), and the points in \( R \setminus R' \) are outside \( C \) as well as the points in \( R' \setminus R \) are outside \( C' \). Then a triangulation is Delaunay iff every pair of opposite d-cells is locally Delaunay. This property has been first noted by Rajan \([145]\). It follows from the fact that the Delaunay triangulation is the unique triangulation having empty circumspheres.

**P6** Let \( \lambda : E^d \to E^{d+1} \) be the following function:

\[
\lambda((x_1, \ldots, x_d)) = (x_1, \ldots, x_d, x_{d+1} = x_1^2 + \cdots + x_d^2)
\]

Let \( H \) be the convex hull of \( \lambda(S) \) and \( DT \) the Delaunay triangulation of \( S \). Then \( DT \) is the projection of lower faces of \( H \) onto the hyperplane \( h_0 : \{ x|x_{d+1} = 0 \} \). A face of \( H \) is a lower face if it has a nonvertical supporting hyperplane \( h \) so that all points in \( S \) lie above or on \( h \).

The function \( \lambda \) maps the space \( E^d \) to the paraboloid of revolution \( \Lambda \) in \( E^{d+1} \). The crucial property of this paraboloid is that any nonvertical hyperplane \( h \) either misses it, touches it at a tangent point or the intersection of \( h \) and \( \Lambda \) projects to a hypersphere in \( E^d \). In this case the portion of \( \Lambda \) below the hyperplane \( h \) projects to the inside of the hypersphere and the portion above to the outside. So for a lower face \( f = \text{chull}(\lambda(R)) \) of \( H \), delimited by the projection of the set of vertices \( R \), there is some nonvertical supporting hyperplane \( h \) with \( \lambda(R) \subset h \) and \( \lambda(S \setminus R) \) above \( h \). The intersection \( h \cap \Lambda \) contains \( \lambda(R) \) and its projection to the hyperplane \( h_0 \) is a hypersphere. Hence the sites in \( R \) lie on a common hypersphere and since \( \lambda(S \setminus R) \) lies above \( h \) the sites in \( S \setminus R \) lie outside this hypersphere. Hence \( \text{chull}(R) \) is Delaunay.
3.2 Voronoi diagrams and Delaunay triangulations

3.2.3 The combinatorial complexity of Voronoi diagrams and Delaunay triangulations

It is important to know the combinatorial complexity of Voronoi diagrams and Delaunay triangulations in order to get a feeling about the best performance a construction algorithm can achieve in the worst case. Since Voronoi diagrams and Delaunay triangulations are dual representations it is sufficient to determine the combinatorial complexity of one of them.

In 2-dimensional space the fact that the Delaunay triangulation is a planar graph allows us to use Euler's formula to obtain the maximal number of 2n-4 Delaunay faces and 3n-6 edges for a triangulation of n sites. Hence the Voronoi diagram in \( E^2 \) can be stored in \( O(n) \) space.

In higher dimensions property P6 is the key to determine the combinatorial complexity of Delaunay triangulations and Voronoi diagrams. It relates the two constructs to (d+1)-dimensional convex polytopes for which there are lower and upper bound theorems established by Brondsted. The upper bound theorem states that the number of j-dimensional faces of a d-dimensional convex polytope with n faces is asymptotically \( O(n^{\min(d-j,\lfloor d/2\rfloor)}) \). We know that the Voronoi diagram of n sites has n d-dimensional Voronoi cells which correspond to n faces of the (d+1)-dimensional polytope to which the Voronoi diagram is related. Hence the number of j-dimensional faces of a d-dimensional Voronoi diagram is asymptotically at most \( O(n^{\min(d+1-j,\lfloor d/2\rfloor)}) \) and therefore the combinatorial complexity of the Voronoi diagram \( O(n^{\lfloor d/2\rfloor}) \). Using known bounds on the number of faces of a convex polytope Seidel [159] has refined this result to give exact bounds for the worst-case number of j-dimensional faces of Voronoi diagrams in d dimensions for all j and d.

For us it is sufficient to know that a worst-case optimal algorithm has a complexity of \( O(n^{d/2}) \).

Since worst-case complexity is seldom achieved, the combinatorial complexity of Voronoi diagrams of randomly distributed sites has been studied. Meijering for example derived means for the volume, total boundary area and the total edge length of Voronoi cells in \( E^3 \), as well as the average number of vertices, edges and faces for a set of point-sites given by a unit-intensity Poisson process. He also showed that for such a site distribution the expected number of neighboring Voronoi cells is about 6 in 2D and 15.54 in 3D. This result can be understood if one notices that inside a uniform point distribution the probability for large empty hyperspheres is very small. Hence point sites probably have only a small number of nearby neighbors. The situation is different at the boundary of point site distributions. Nevertheless Dwyer [54] showed that the expected number of vertices of a d-dimensional Voronoi diagram is only \( O(n) \) if the n sites are uniformly distributed in a hypersphere. He further conjectures that the bounds hold for uniformly distributed point sites in an arbitrary convex polytope. This results are true because if only a small fraction of the points lie near the boundary edge effects are negligible. At this point we have to note that in our case all the point sites lie on a common surface. This means that we cannot use any of these results. On the contrary our Voronoi diagrams will probably have a complexity above the average.

3.2.4 Algorithms constructing Voronoi diagrams and Delaunay triangulations

The brute-force algorithm to construct the Delaunay triangulation would be the following:

1. Compute for all (d+1)-tuples of sites the circumscribing hypersphere and its radius
2. Accept all those tuples which have no other sites lying in their hypersphere as Delaunay cells.

This algorithm however has a complexity of \( O(n^{d+1}) \) which makes it impractical. Hence there have been enormous efforts to device better algorithms. In this section we would like to review them according to five basic types of construction manners and to their historical evolution. These types of construction are:

- local transformation of an arbitrary triangulation
- incremental insertion of point sites and updating an existing Delaunay triangulation
3.2 Voronoi diagrams and Delaunay triangulations

- incremental construction of the Delaunay triangulation cell by cell using the empty circumsphere property
- divide & conquer methods using recursive partitioning of the point set and merging the local triangulations to a global one
- higher dimensional embedding of the point sites and backprojecting their convex hull

There is another algorithm, the so-called plane-sweep algorithm, proposed by Preparata and Shamos [142] which doesn’t really fit into one of these categories. It is very efficient in 2D and could also be generalized to 3D. However its performance essentially relies on the planarity of the 2D Voronoi diagram. Hence its generalization to 3D can not be hoped to produce an efficient Voronoi diagram computation scheme and thus we will not review it.

Local transformation method

Lawson [99] was the first to use local transformations to construct the Delaunay triangulation in 2D. He used the equiangularity property which is an equivalent formulation for 2D of the more general property P5. The algorithm can be sketched as follows:

1. determine an arbitrary triangulation of \( S \)
2. as long as there are two opposite triangles \( abc \) and \( acd \) forming a convex quadrilateral \( abcd \) for which the minimum of the six angles of the triangles \( abc \) and \( acd \) is less than the minimum of the six angles of the alternative triangles \( abd \) and \( bed \) flip the edge \( ac \) to the alternative edge \( bd \) (see figure 3.8).

![Figure 3.8: The diagonal edge of the quadrilateral formed by two adjacent triangles is flipped whenever this leads to a larger minimal angle](image)

It can be shown that an edge which once has been disappeared by an edge flip can never show up again in a subsequent triangulation. Together with the fact that the common edge of two triangles forming a nonconvex quadrilateral is already locally Delaunay this algorithm is guaranteed to converge to the Delaunay triangulation after at most \( \binom{n}{2} \) (the number of possible edges between \( n \) sites) edge flips. Thus the algorithm has \( O(n^2) \) worst-case complexity. This complexity can however be reduced to \( O(n) \) if the point sites are inserted incrementally
3.2 Voronoi diagrams and Delaunay triangulations

in random order (cf. [69]). The algorithm has been generalized to 3D using property P5 by Joe [84, 85]. This is not straightforward since in 3D 5 points can form more than two tetrahedra and the flipping procedure can get stuck as Joe ([84]) has noted. However by starting with a special triangulation it is always possible to obtain a Delaunay triangulation by a finite sequence of local transformations in an appropriate order (see [85]). His algorithm has a worst-case time complexity of $O(n^2)$ which is optimal. Further he gives an empirical time complexity for uniformly distributed sites of $O(n(\log n)^2)$.

**Incremental insertion**

Incremental insertion of point sites into an existing Delaunay triangulation or Voronoi diagram is the most popular construction method. It has been proposed for the first time by Green and Sibson [68] and stands out for its simplicity. Depending on whether the sites are known a priori or not the method is initialized either by a d-dimensional polytope containing all the sites and formed by $d+1$ artificially introduced sites or simply by the d-dimensional polytope formed by the first $d+1$ known sites (if they are not in degenerate position, i.e. lying on a common (d-1)-dimensional hyperplane). Then inserting a new site involves two steps:

1. Locate all d-dimensional Delaunay cells which are affected by the insertion, i.e. have the new site in their hypersphere. Together they form a d-dimensional polytope.
2. Retriangulate the polytope from the first step.

In 2D the incremental algorithm is easy and has been described by Green and Sibson [68]. The first step can be reduced to a nearest neighbor search which can be performed in $O(\sqrt{n})$ time on the previously computed Delaunay triangulation. Due to property P1 we know that the perpendicular bisector between the new site and its nearest neighbor is a Voronoi edge. Intersecting this new edge with the other edges of the nearest neighbor’s Voronoi cell one finds the next neighboring site which together with the new site determines the next Voronoi edge of the new Voronoi cell. The process continues until the new Voronoi cell is completed. In the worst case the first step is of complexity $O(i)$ with $i$ denoting the number of sites so far inserted. The second step, if appropriately implemented, requires time proportional to the number of new Voronoi edges generated. In the worst case this is again $O(i)$. Hence the overall complexity is $\sum_{i=1}^{n} 2i = 2 \cdot \frac{n^2+3n+2}{2} = k \cdot (n^2 + n - 6) = O(n^2)$. A variant of this algorithm working on the Delaunay triangulation is given by Lee and Schachter [102] as well as by Guibas and Stolfi [70]. In the first step the triangle in which the new site lies is located and the new site is connected with the vertices of this triangle. This gives up to four (in the case the new site lies on a Delaunay edge) convex quadrilaterals whose diagonal can be swapped if it does not fulfill the equiangularity property. This algorithm is an incremental insertion variant of the local transformation algorithms. It has again an $O(n^2)$ complexity. Since worst-case complexity is seldom achieved the runtime complexity for special distributions of the input sites is of much more interest. The algorithms as described above have an $O(n^{3/2})$ complexity. Due to Ohya, Iri and Murota [131, 132] the expected time for finding the region the next site falls in and for determining the new region can be reduced to $O(1)$ by introducing a suitable ordering of the sites using a uniform grid structured as a quaternery tree. This gives an expected runtime of $O(n)$ for a variety of site distributions. Another possibility is the insertion of sites in random order as proposed by Guibas, Knuth and Sharir [69]. They show that with randomized insertion order the expected total number of structural changes in updating the triangulation is of $O(n)$ regardless of the site’s distribution. Hence the first step becomes the critical part of the algorithm. The authors show that the expected overall time to locate for each newly inserted site the triangle into which it falls is $O(n \log n)$ time. It is sufficient to retain all the previous triangulations which can be done in $O(n)$ space in a structure similar to the Delaunay tree proposed by Boissonat and Teillaud [24]. A new site’s location relative to the actual triangulation can then be determined by tracing all triangles containing the new site in the chronological order of their creation. Thereby the first triangle is the one enclosing the whole triangulation.
Already in 1981 two incremental insertion algorithms in higher-dimensional space have been proposed. One by Watson [184] based on the Delaunay triangulation and one by Bowyer [28] based on the Voronoi diagram. In Watson’s algorithm the point sites are inserted in an 'advancing front' sequence. The Delaunay cells behind the front, i.e. those having a smaller hypersphere radius than distance to the front, are completed while those ahead of the front are subject to alteration. The first step is then implemented as a sequential search on all alterable Delaunay cells to find those having the new site in their hypersphere. All those Delaunay cells form a d-dimensional polytope. The (d-1)-dimensional faces inside this polytope are then deleted and with each boundary face a new Delaunay cell is formed. Watson claims that this algorithm has a time complexity of \( O(n^{2-{\frac{1}{d}}}) \) for well distributed sites. An implementation of Watson’s algorithm in 3D is discussed by Field [62]. He particularly discusses how one has to cope with degenerated point sets coming from regular samplings of solid models. In Bowyer’s algorithm the sites are inserted randomly. In the first step a Voronoi vertex \( v \) is identified which will be deleted\(^3\) by the insertion of the new site. Such a vertex is any that is nearer to the new site than to any of its forming sites. It can be found by walking through the Delaunay structure from neighbor to neighbor starting at a site near the centroid of the already inserted sites. Bowyer conjectures that this takes \( O(n^{\frac{1}{d}}) \) time for well distributed sites. From the Voronoi vertex \( v \) all other vertices which will be deleted can then be found in a tree search on the Voronoi diagram. The set of sites which formed these deleted Voronoi vertices are then the sites which together with the new site form the new Voronoi cell which can then be computed. Bowyer claims that his algorithm has \( O(n^{1+\frac{1}{d}}) \) time complexity provided the point sites are well distributed. The algorithms of Lee and Schachter [102] and Guibas and Stolfi [70] which uses the equiangularity property has been generalized to d dimensions by Rajan [145] by providing property P5 which is related to the equiangularity but generalizes to d dimensions. Sugihara and Inagaki [172] describe a method to make the incremental insertion algorithm robust against degeneracies. They additionally report that the first step of incremental insertion can be performed in constant time in the average if a uniform grid is used to speed up point location. Hence the combinatorial complexity comes down to \( O(n) \) for the average case of well distributed point sets.

Incremental insertion methods are well suited when the point sites are not a priori known or when they change over time. They can be easily stabilized with respect to degeneracies in 2D and, with a bit more overhead, in 3D, too. Simple implementations have an average-case time complexity of \( O(n^{1+\frac{1}{d}}) \). They can be improved to yield as less as \( O(n) \) computation time for special site distributions.

### Incremental Construction

Incremental construction methods directly generate the final Delaunay triangulation and/or Voronoi diagram from an a priori known set of point sites. They use all the empty circumsphere property to find either the Delaunay cells or the Voronoi vertices or both. They are incremental in the sense that they begin to construct e.g. the Delaunay triangulation with an arbitrary Delaunay cell and then incrementally search for neighboring Delaunay cells. They rely on the fact that each (d-1)-face inside the Delaunay triangulation has exactly two d-cells. Hence they fit into the following scheme:

1. Find one Delaunay cell and use it to initialize a face list
2. As long as there are faces in the face list find for each face the missing Delaunay cell and update the face list, i.e. for all faces \( f_i \) of the new Delaunay cell:
   (a) if \( f_i \) is already in the face list, remove it.
   (b) otherwise add \( f_i \) to the face list.

An early approach to incremental construction in the plane has been proposed by McLain [115]. Given an edge \( s_0s_1 \) of a triangle \( s_0s_1s_2 \) he computes for all sites \( s_i \) lying with respect to \( s_2 \) on

---

\(^3\)Note that an old Voronoi vertex will only be deleted if the new site is inserted inside the convex hull of the previous points.
3.2 Voronoi diagrams and Delaunay triangulations

the other side of \( s_0s_1 \) the radius of the circumcircle of the triangle \( s_0s_1s_i \). The triangle with least radius is then accepted as a new Delaunay triangle. Clearly this algorithm has a worst-case complexity of \( O(n^2) \). An improved version which has \( O(n) \) expected running time for uniformly distributed sites is due to Maus [113]. The speedup is achieved by a scheme for fast point location: the uniform grid. This technique is well known and widely used in geometric applications. We will describe it in detail in section 3.3. The uniform grid partitions the plane into small boxes. Obviously the circumcenter of the new Delaunay triangle must lie somewhere on the bisector of \( s_0s_1 \). This fact together with the fine-grained partitioning of the plane provided by the uniform grid can be used to design a fast search-strategy for the third point of the new Delaunay triangle. In fact it is sufficient to search only in the boxes which are

(a) lying on the other side of \( s_0s_1 \) than \( s_2 \) and

(b) fully or partly contained in an ever larger circle passing through \( s_0 \) and \( s_1 \) and having its center on the bisector of \( s_0s_1 \).

Hence in Maus’ algorithm only a minimal part of the plane is searched for a site to find the new Delaunay triangle. Recently the same method has been reproposed and described in a more detailed manner by Fang and Piegl [59].

In 3D two early methods have been proposed which do not use the above scheme but directly construct Voronoi polyhedra. Both have a complexity of \( O(n^2 \log n) \) for uniformly distributed sites. The one of Brostow, Dussault and Fox [31] computes for each point site \( s \) the Voronoi polyhedron \( V(s) \) face by face. It consists of the following three steps:

1. finding a set of direct neighbors of \( s \). A direct neighbor \( s_d \) of \( s \) has the property that the midpoint of \( ss_d \) is contained in the Voronoi polyhedron \( V(s) \).

2. constructing a polyhedron with the direct neighbors.

3. refine the polyhedron to the Voronoi polyhedron \( V(s) \) by adding the additional neighboring sites.

The one of Finney [63] computes first the Voronoi vertices and then derives the edges and faces of the Voronoi polyhedra. Finney’s method has the problem that it relies on two empirically determined spheres \( S_c \) and \( S_s \) around each site. For each site \( s_i \) three other sites inside the sphere \( S_c \) around \( s_i \) are searched. They form a combination of four sites defining a candidate for a Voronoi vertex. To determine whether this candidate vertex is a Voronoi vertex or not it is checked whether the sites inside sphere \( S_c \) are nearer to the vertex or not. However in a general case it is difficult to determine in advance such spheres which can guarantee correct results. Besides that, the efficiency of the algorithm depends on the selection of these spheres. Tanemura, Ogawa and Ogita [179] describe the construction of a single Voronoi polyhedron by finding all the Delaunay polyhedra corresponding to the vertices of the Voronoi polyhedron. It was one of the first using the above mentioned scheme of incremental construction in 3D. The disadvantages of their algorithm are that they construct the Voronoi polyhedron of a site \( s \) assuming that only the \( n_s \) nearest point sites (in their case \( n_s \approx 40 \)) are involved in the formation of the polyhedron and that they construct every Voronoi polyhedron independently. The first problem is attenuated since they perform a consistency check on the Voronoi polyhedron. Inconsistent polyhedra could be newly constructed with an increased value of \( n_s \). Contemporarily Avis and Battacharya [13] have given an algorithm which determines the Voronoi vertices and edges of d-dimensional Voronoi diagrams in \( O(n^{\frac{d}{2}}+1) \) time. Given a Voronoi vertex and its \( d+1 \) defining Delaunay vertices, all \( \binom{d+1}{1} \) subsets of \( d \) vertices are generated. Each of them represents a Voronoi edge on which the next Voronoi vertex can be found by searching a new additional Delaunay vertex to make again a complete set of \( d+1 \) Delaunay vertices from the subset. All Voronoi edges which have been found for the second time are marked. The procedure terminates when no more unmarked Voronoi edges exist. Their second approach constructs the Delaunay triangulation by using the fact that the \( s_i \) is a Delaunay edge if and only if \( H(s_i, s_j) \)
is a non-redundant halfspace in the intersection \( \cap_{k \neq l} H(p_l, p_k) \). Since halfspaces can be described by inequalities, the redundancy of a halfspace can be determined by linear programming. The give no complexity analysis on this algorithm. A more efficient algorithm which computes higher-dimensional Voronoi diagrams in linear expected time for uniformly distributed sites has been presented by Dwyer [54]. Dwyer makes a careful complexity analysis of his algorithm. It is similar to Maus' planar algorithm. A slight improvement is the use of a priority queue into which the boxes to be searched are inserted. In the queue the boxes are ordered by the minimal circumsphere radius that can be achieved by a point lying inside the box (for computational reasons all points inside the smallest sphere fully containing the box are taken into account). Essentially the same algorithm has been described recently by Fang and Piegl [60]. They have another strategy of searching the boxes and make some additional considerations about completeness and correctness of their algorithm. Without further analysis they claim based on empirical results that it runs in \( O(n) \) time, too.

### Divide & Conquer (D&C) methods

The first worst-case optimal algorithm for Voronoi diagrams in 2D used the D&C method. It has been proposed by Shamos and Hoey [164]. D&C methods split the problem at hand into smaller subproblems, compute their solutions recursively (unless they are very small and can be solved by trivial methods), and finally combine the partial solutions to the global one. Applied to the Voronoi diagram computation this means that first the underlying set \( S \) of sites is partitioned into two subsets \( S_1 \) and \( S_2 \) of nearly equal cardinality. Then the Voronoi diagrams \( V_1(S_1) \) and \( V_2(S_2) \) are computed recursively and merged to the global Voronoi diagram \( V(S) \). The merge step deserves the main attention since it actually constructs the diagram. In the following short description of it we assume that \( S_1 \) and \( S_2 \) are separable by a vertical line. Every point \( p \) of the plane falls into the Voronoi cell \( V_1(s_i) \) for some site \( s_i \in S_1 \) and also into the Voronoi cell \( V_2(s_j) \) for some site \( s_j \in S_2 \). The condition that \( p \in V(s_i) \) of \( V(S) \) is: \( \delta(p, s_i) < \delta(p, s_j) \). Otherwise \( p \in V(s_j) \). So we have to cut off some part from \( V_1(s_i) \) and from \( V_2(s_j) \) by means of the bisector of \( s_i \) and \( s_j \) in order to get \( V(s_i) \) and \( V(s_j) \). This is exactly what the merge process has to do. Other authors [102, 53, 70] have described D&C methods to construct the Delaunay triangulation in the plane. Among them Guibas and Stolfi [70] give a careful and detailed description. If the set of sites is uniformly distributed in the unit square, then the expected runtime can be improved. Bentley, Weide and Yao [14] show that a combination of bucketing techniques and any \( O(n \log n) \) algorithm leads to an algorithm with expected runtime \( O(n) \). Dwyer [53] shows that a simpler algorithm has an expected runtime of \( O(n \log \log n) \) for a large class of input site distributions. His algorithm first splits the plane into square cells, then computes the triangulation inside the cells with the Guibas-Stolfi algorithm and finally merges the cells first to rows and then to the rows to the whole plane.

The generalization of the D&C algorithm to higher dimension is difficult and in the literature no general D&C solution to the Voronoi diagram construction has been proposed up to now. The problem is the design of an efficient merge phase. In 2D it relies on the fact that the edges incident in a vertex can be explicitly ordered. In higher-dimensional space this ordering is not given. Nevertheless a solution to the problem has been described by Cignoni et al. [42] which adopts a slightly modified D&C concept. It reverses the order between the subproblems solution and the merging phase. Since we have chosen to follow this algorithm we defer its detailed description to section 3.3.

### Higher-dimensional embedding

Already in 1979 Brown recognized the relation between Voronoi diagrams in 2D and convex hulls in 3D. The algorithm makes use of a geometric transform called inversion. The inversion transform is determined by two parameters:

1. the center of inversion denoted by \( P_I \)
2. the radius of inversion denoted by \( r_I \)
When we chose the center of inversion to be the origin of the coordinate system and the radius of inversion to be one, then the inversion transform is defined by the following mapping, given in polar coordinates \((r, \theta, \phi)\):

\[
f_I : (r, \theta, \phi) \mapsto \left(\frac{1}{r}, \theta, \phi\right)
\]

Inversion in general has the following two important properties:

1. it is involutory, i.e. applied twice to a point yields the original point
2. a sphere which passes through the center of inversion \(P_I\) transforms to a plane which does not pass through \(P_I\) and the interior of the sphere is mapped to the halfspace of the plane that does not contain \(P_I\) and the exterior to the other halfspace.

The Voronoi diagram in 2D can then be constructed as follows:

1. for a set of input sites in the xy-plane, chose the center of inversion \(P_I\) in the space above the plane. The inversion transforms the xy-plane to a sphere passing through the center of inversion. Hence it lifts the input sites onto a sphere in 3D.
2. determine the convex hull of the lifted point set. The faces of the convex hull define again planes which by a second inversion transform are mapped to spheres. These spheres intersect the xy-plane in circles. The input sites which defined the convex hull face will lie on the intersection circle.
3. Now note that all points lying on the same side of the convex hull face as the inversion center \(P_I\) will lie outside the sphere to which the face is transformed. Hence all faces of the convex hull which have the center of inversion \(P_I\) on the same side as the rest of the convex hull will be transformed to a sphere inside which there are no other sites. Therefore the intersection circle of this sphere with the xy-plane will also contain no other site. It is the empty circumcircle of the sites lying on the circle.

The time consuming part of this algorithm is clearly the determination of the convex hull. Methods to compute convex hulls have been proposed by Preparata and Hong [141] in 2D and 3D. It requires \(O(n \log n)\) time. For higher dimensions Chand and Kapur [39] presented the so-called 'gift-wrapping' algorithm which has a worst-case runtime complexity of \(O(n^d+1)\). A newer technique is called the beneath-beyond method and is described by Preparata and Shamos [142]. It has also runtime complexity \(O(n^d+1)\). A randomized algorithm with expected runtime of \(O(n^{d+\frac{1}{2}})\) is due to Seidel [160, 161].

Brown's idea was taken up by Edelsbrunner and Seidel [57, 55]. They defined a geometric transform \(\pi\) which maps each site \(s = (s_1, \ldots, s_d)\) in \(R^d\) to a hyperplane \(\pi(s)\) in \(R^{d+1}\). The transform \(\pi\) is defined as follows.

\[
\pi(s) : x_{d+1} = 2s_1x_1 + \cdots + 2s_dx_d - (s_1^2 + \cdots + s_d^2)
\]

As one can see in figure 3.9 \(\pi(s)\) is the unique hyperplane that touches the unit paraboloid

\[
U : x_{d+1} = x_1^2 + \cdots + x_d^2
\]

in the vertical projection \(U(s)\) of \(s\) onto \(U\). The transform \(\pi\) translates distance information from \(R^d\) into combinatorial information in \(R^{d+1}\) as we can see from the following relation for two points \(s\) and \(x\) in \(R^d\):

Let \(h\) be the hyperplane defined by transform \(\pi\) for site \(s\): \(h = \pi(s)\) and \(U\) be the paraboloid as defined above. Then the following equation holds

\[
\delta^2(x, s) = \delta(U(x), h(x))
\]

To verify this equation it is sufficient to substitute \(U(x)\) and \(h(x)\) with the equations given above. Now we can relate the Voronoi diagram \(V(S)\) for a set \(S\) of input sites to the upper enveloppe
3.2 Voronoi diagrams and Delaunay triangulations

Figure 3.9: The transform \( \pi \) relates the input sites to hyperplanes. The upper envelope (hatched region) of them backprojected to the lower-dimensional space yields the Voronoi diagram.

Degeneracies and numerical errors

Geometric algorithms, when implemented, often fail due to the degeneracies in the input data and to numerical errors introduced by the finite precision arithmetic computations. In general, these algorithms deal with two types of data – numerical and combinatorial. Combinatorial inferences such as face adjacencies and vertex adjacencies are derived from the numerical data. Thus, inaccuracies in numerical computations may cause inconsistencies in combinatorial data which in effect either produces invalid output or makes the program fail. Various approaches have been proposed to cope with the problem.

A plausible approach is to use integers instead of reals. This seems to be very attractive for us since our input sites come from the integer raster grid. However already a coarse estimate of the bit complexity for the computation of Voronoi vertices shows the impracticability of exact arithmetic. The computation of a Voronoi vertex in \( d \) dimensions can be done by evaluating the determinant of a \( d+2 \)-dimensional matrix. Viewing the determinant as a polynomial we can give a rough estimate of the bit complexity of its evaluation. If a polynomial has degree \( d \) and the variables have bit length \( b \), then approximately \( bd \) bits are required to evaluate the polynomial. Hence with conventional 32-bit integer arithmetic the input coordinates in dimension \( d = 3 \) could have a maximal bit length \( b = 5 \) plus one sign bit. This means that the input coordinates would have to lie on a \( 32 \times 32 \times 32 \) grid. Therefore arbitrary-precision integer arithmetic or at least extended-precision integer arithmetic would be required and lead to a serious loss in efficiency.
3.3 Computation of the Delaunay Triangulation

As we have seen many algorithms for 3D VD generation have been proposed in computational geometry and some implementations are available. Most of them, however, run into serious
problems when faced with such a large number of generating points. In addition, they are usually unable to handle degenerate cases caused by more than four cospherical points; a situation that is unavoidable among point samples located on a regular image raster. Handling cospherical points calls for more flexible data structures to represent a DT consisting not only of tetrahedra but of general convex polyhedra.

In our implementation we extended an algorithm proposed by Cignoni et al. [42] adapting it to deal with cospherical input points and supplying it with a uniform grid for fast point location, a technique which is well known from computer graphics applications. The algorithm actually computes the DT of a point set which is the dual representation of the VD. It is based on the divide & conquer (D&C) paradigm, which basically consists in recursively applying two phases: a subdivision of the problem into subproblems and a merging of the subproblems' solutions. The efficiency of the D&C algorithm depends on an efficient merging of the local DT's computed in the previous phase. The merging requires a number of local modifications on both DT's which is an expensive task. This can be avoided if instead of merging partial results in a second step, the merging part of the DT is built first and then the independent parts of the DT are computed on subsets of the input points.

![Figure 3.10: Construction of an initial Delaunay tetrahedron using properties P1 - P3](image)

We have chosen to follow Cignoni's algorithm both for its time characteristics (it can be parallelized and has been proved to be optimal in 2D both in terms of mean and worst time complexity) and its memory requirements: the Delaunay polyhedra can be stored to disk immediately when they are generated and only a relatively small list of Delaunay faces have to be retained in memory.

In the following we will give a detailed description of our implementation.

Let us first begin with the two basic functions used by incremental construction algorithms:

[F1 ] construction of an initial Delaunay polyhedron

[F2 ] construction of a Delaunay polyhedron given a Delaunay face and the side on which the missing point(s) must lie

The first function [F1] finds a Delaunay polyhedron vertex by vertex. One can arbitrarily choose a first vertex \(v_1\). Then the nearest neighbor of \(v_1\) is taken as second vertex \(v_2\). The edge \(v_1v_2\) is a Delaunay edge thanks to property P1 (cf. section 3.2.2). The third vertex \(v_3\) is determined so that the circumcircle of the triangle \(v_1v_2v_3\) is minimal which guarantees that \(v_1v_2v_3\) is a Delaunay face due to property P2. Finally, the forth vertex \(v_4\) is defined so that the tetrahedron \(v_1v_2v_3v_4\) has a minimal circumsphere and will be a Delaunay tetrahedron thanks to property P3. The construction of this initial Delaunay tetrahedron is illustrated in figure 3.10. This would be sufficient if the pointset didn't contain cospherical sites. In our case however an additional check must be performed to find cospherical points. We call this function FirstPolyhedron() and give the pseudocode for it in listing 3.11.
FirstPolyhedron() needs the function circumcircle() to compute the center of the circumcircle of three points and the function circumsphere() to compute the center of the circumsphere of four points.

The pseudocode for the determination of the circumcircle of three points \( v_0, v_1 \) and \( v_2 \) is given in listing 3.12. The center of the point's circumcircle can be computed as the intersection of two bisecting planes with the plane defined by the three points. A bisecting plane \( h \) is defined by two vertices \( v_0 \) and \( v_1 \). It subdivides the space into two halfspaces \( H^+, H^- \) each of which contains the part of space lying nearer to one of the two vertices. It is defined by the normal vector \( \mathbf{n}_{vo,v1} \) (cf. step 1) and the point \( \frac{v_0 + v_1}{2} \). If \( v_0, v_1 \) and \( v_2 \) are collinear they do not define a unique plane. Hence the normal vector computed in step 1 is near zero and the function terminates. Since in real world all computations are made in finite precision we must supply an \( \epsilon \) for this equality test. If \( v_0, v_2 \) and \( v_2 \) are in non-degenerate position we have three equations, one for each plane. This leads to the system \( Ax = b \). The solution can be computed efficiently according to Cramer's rule. \( A_i \) designates the matrix \( A \) for which the \( i \)-th column has been replaced by the vector \( b \), \( x_i \) the \( i \)-th coordinate of the solution:

\[
x_i = \frac{\det(A_i)}{\det(A)}
\]

The pseudocode for the determination of the circumsphere of four points is given in listing 3.13. From the use we will make of this function later we assume that as input are given:

- a face \( f = \{v_0, v_1, v_2\} \) (i.e. a triangle or the first three points of a polygon)
- its normal \( \mathbf{n}_f \)
- the center \( c_f \) of the face's circumcircle
- a fourth vertex \( v_3 \)

The four points \( f \cup v_3 \) do not define a tetrahedron if they are coplanar. In this case the vector product of the face's normal with a vector from a point \( p \) on the face \( f \) to \( v_3 \) is near zero (cf. step 2). Since the center \( c_f \) of the face's circumsphere is already known we can use it to compute efficiently the center \( c_{DP} \) of the tetrahedron's circumsphere. Thereby we use the fact that \( c_{DP} \) must lie on the line passing through \( c_f \) in the direction of the face's normal. The equation is given in step 3. One can easily see that the point \( c_f \) can be replaced by any point on the line \( c_f + \alpha \mathbf{n}_f \). We will use this fact in the implementation of function [F2] where we do not know the face's circumcenter but the second tetrahedron of the face (if it is not on the convex hull).

```c
function FirstPolyhedron(P)

1. \( v_0 = \text{random}(p_i) \)
2. \( v_1 = p_i, \min (\delta(v_0, p_i)) \)
3. \( v_2 = p_i, \min (\delta(v_0, c_i)), c_i = \text{circumcircle}(v_0, v_1, p_i) \)
   \[ c_f = \text{circumcircle}(v_0, v_1, v_2) \]
   \[ f = \{v_0, v_1, v_2\}, \mathbf{n}_f = \frac{v_0 v_1}{v_0 v_2} \times v_0 v_2 \]
4. \( v_3 = p_i, \min (\delta(v_0, c_i)), c_i = \text{circumsphere}(f, \mathbf{n}_f, c_f, p_i) \)
   \[ c_{DP} = \text{circumsphere}(f, \mathbf{n}_f, c_f, v_3) \]
5. \( \text{cospherical.vrts} = \{p_i \neq v_0, v_1, v_2, v_3 \mid \delta(p_i, c_{DP}) = \delta(v_0, c_{DP}) + \epsilon\} \]
6. RETURN \{4+|\text{cospherical.vrts}|, v_0, v_1, v_2, v_3, \text{cospherical.vrts}\}
```

Figure 3.11: Pseudocode of the determination of the first polyhedron
function circumcircle(v₀, v₁, v₂)

1. \( \vec{n}_0 = v_0 \vec{v}_1 \)
2. \( \vec{n}_1 = v_0 \vec{v}_2 \)
3. \( \vec{n}_2 = \vec{n}_0 \times \vec{n}_1 \)
4. IF \( |\vec{n}_2| < \epsilon \) THEN RETURN colinear.vrts
5. \( \vec{b} = \begin{pmatrix} v_0 \\ v_1 \\ \frac{v_0 + v_1}{2} \\ \frac{v_0 + v_1}{2} \\ \vec{n}_2 \vec{v}_0 \end{pmatrix} \)
6. Solve \( A \vec{x} = \vec{b} \)
7. RETURN \( \vec{x} \)

Figure 3.12: Pseudocode of the determination of the center of a triangle's circumcircle

function circumsphere(\( f = \{v_0, v_1, v_2\}, \vec{n}_f, c_f, v_3 \))

1. \( \vec{n}_0 = v_0 \vec{v}_3 \)
2. IF \( |\vec{n}_f \times \vec{n}_0| < \epsilon \) THEN RETURN coplanar.vrts
3. \( \vec{c}_{DP} = \vec{c}_f + \left( \frac{(\vec{n}_f \times \vec{n}_0) \times v_0}{\vec{n}_f \times \vec{n}_0} \right) \times \vec{n}_f \)
4. RETURN \( \vec{c}_{DP} \)

Figure 3.13: Pseudocode of the determination of the center of a tetrahedron's circumsphere

The second function \([F2]\) is called NextPolyhedron(). Given a face \( F = \{v_{i0}, v_{i1}, v_{i2}\} \) of a Delaunay polyhedron \( DP_i = \{v_{i0}, v_{i1}, v_{i2}, v_{i3}, \ldots, v_{ik}\} \) it determines the neighboring polyhedron \( DP_j = \{v_{j0}, v_{j1}, v_{j2}, v_{j3}, \ldots, v_{jk}\} \). Note that a Delaunay face always makes part of exactly two polyhedra unless it is part of the pointset's convex hull. Hence the second polyhedron can be found by searching for a fourth vertex \( v_{j3} \) which must lie in the halfspace of \( f \) that doesn't contain the vertices of the first, already known polyhedron. We denote this halfspace by \( H_f^{v_{i3}} \) \((-v_{i3} \text{ because it does not contain } v_{i3})\) and the opposite by \( H_f^{+v_{i3}} \). During the computation we assign a side information to each Delaunay face at the time when only one polyhedron to which the face belongs is known. The side information is used later to determine whether a point \( p_i \) lies in the correct halfspace or not. On which side a point \( p_i \) lies can be defined by the angle \( \phi_{p_i} \) between the face's normal \( \vec{n}_f \) and the vector \( \overrightarrow{v_{i0}p_i} \) from a vertex \( v_{i0} \in f \) to the point \( p_i \). Let us assume that the face's normal points into the halfspace \( H_f^{+v_{i3}} \). Then according to the location of \( p_i \) the angle \( \phi_{p_i} \) falls into one of three possible intervals:

1. \( \phi_{p_i} < 90^\circ \) if \( p_i \) lies in the halfspace \( H_f^{+v_{i3}} \).
2. \( \phi_{p_i} = 90^\circ \) if \( p_i \) lies on the plane defined by \( F \).
3. \( 90^\circ < \phi_{p_i} < 180^\circ \) if \( p_i \) lies in the halfspace \( H_f^{-v_{i3}} \).
The cosine of $\phi_p$ equals $\vec{n}_f \cdot \vec{v}_0 \vec{p}_i$ and is positive in the first case, zero in the second and negative in the third. As side information of the face $f$ we use the sign of $\cos(\phi_{v_13})$ where $\phi_{v_13}$ is the angle determined for a vertex $v_{13}$ belonging to the vertices of the already known polyhedron but not to the face $f$. Then a point $p_i$ lies not on the same side of the face as $v_{13}$ (independently of the orientation of the face’s normal!) if the following condition holds for the angle $\phi_{p_i}$:

$$\text{sgn}(\cos(\phi_{v_13})) \cdot \text{sgn}(\cos(\phi_{p_i})) < 0$$

In step 3 we evaluate for all the points $p_i$ which are lying on the desired side of the face the circumsphere of the polyhedron $\{v_{i_0}, v_{i_1}, v_{i_2}, p_i\}$. This gives us a center $c_{p_i}$ and a radius $r_{p_i}$ for each point $p_i$. We multiply the radii $r_{p_i}$ by $-1$ if the center $c_{p_i}$ lies in the halfspace $H_f^{+v_{13}}$ and call $r_{p_i}$ the signed radius of the circumsphere. Among all the points $p_i$ the one with minimal $r_{p_i}$ is selected. If there was no point on the desired side of the face, the number of vertices returned will be zero (cf. step 5). This is the case when the face is on the convex hull of the pointset. Otherwise, due to property P4, the vertices $v_{i_0}, v_{i_1}, v_{i_2}$ and $p_i$ form a Delaunay tetrahedron. Finally there must be again a check for cospherical points. The pseudocode of the function is given in listing 3.14.

$$\text{function NextPolyhedron}(P, f = \{v_{i_0}, v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}, \vec{n}_f, c_{DP_1}, \text{side})$$

1. FOR each $p_i \in P$ DO
2. $\cos(\phi_{p_i}) = \vec{n}_f \cdot \vec{v}_0 \vec{p}_i$
3. IF ($\text{side} \cdot \text{sgn}(\cos(\phi_{p_i})) < 0$) THEN
   $c_{p_i} = \text{circumsphere}(f, \vec{n}_f, c_{DP_1}, p_i)$
   $r_{p_i} = -\delta(p_i, c_{p_i}) \cdot \text{side} \cdot \text{sgn}(\cos(\phi_{c_{p_i}}))$
ELSE
4. $r_{v_{13}} = \infty$
5. IF ($r_{v_{13}} := \infty$) RETURN {0}
6. cospherical_vrts = \{ $p_i \neq v_{i_0}, v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k} \mid \delta(p_i, c_{v_{13}}) = \delta(v_{i_0}, c_{v_{13}}) + \epsilon$\}
7. RETURN $\{4+|\text{cospherical_vrts}|, v_{i_0}, v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_k}, \text{cospherical_vrts}\}$

**Figure 3.14: Pseudocode of the determination of the next polyhedron**

The definition of an incremental construction of the Delaunay Triangulation following the general scheme given in subsection 3.2.4 is then straightforward. A list of faces $f_{\text{list}}$ is used which holds the Delaunay faces for which only one polyhedron has already been found. Each face $f$ in this list contains the following data for the sake of efficiency:

- the set of its vertices $\{v_{i_0}, \ldots, v_{i_k}\}$
- the face’s normal $\vec{n}_f$
- the center $c_{DP_1}$ of the circumsphere of the Delaunay polyhedron $DP_1$ it belongs to
- the side information $f_{\text{side}}$ as defined above

The face list is initialized with the faces of the first Delaunay polyhedron. Until there are faces in the list, the missing polyhedron is searched for each of them. If the face was not on the convex hull of the pointset a polyhedron is found and the face list is updated by the newly determined faces. The updating of the face list has to handle two different cases:

1. If the face is not yet in the list, it is inserted into it.
2. If the face is already in the list it is removed from it and written to disk.
3.3 Computation of the Delaunay Triangulation

In an object-oriented environment it is natural to model the face list as a class which apart from the standard constructors and destructors has to offer two methods:

- **update(f)**: the face list is updated with face f
- **next()**: a reference to the next face in the list is given

The pseudocode of the procedure \texttt{Triangulate\_Inc\_Constr()} which implements this algorithm is given in listing 3.15. The functions \texttt{FirstPolyhedron()} and \texttt{NextPolyhedron()} both return a list of vertices forming a Delaunay polyhedron. The algorithm then needs the faces of the polyhedron. If the polyhedron has only four vertices (this is always the case if no cospherical points exist in the pointset \( P \)) the faces are exactly the four possible subsets of three vertices. In the general case however, the faces must be determined by a standard convex hull algorithm (designated by \texttt{hull()} in the pseudocode). The procedure \texttt{Triangulate\_Inc\_Constr()} generates the following output:

- A file containing the Delaunay polyhedra. For each polyhedron the vertices are stored.
- A file containing all the Delaunay faces. For each face the vertices and two references to the Delaunay polyhedra having the face in common are stored. If the face is on the convex hull one of the two references is void.
- A file containing the Voronoi vertices, i.e. the centers of the Delaunay polyhedra's circum-spheres.

\begin{verbatim}
procedure Triangulate\_Inc\_Constr(P)
1. f\_list = <new empty list>
2. DP0 = FirstPolyhedron(P)
   write DP0, cDP0 to disk
3. FOR each fi \in F0 DO
   f\_list.update(f_i)
4. WHILE (size(f\_list) > 0) DO
   5. / = f\_list.next()
   6. DP_i = NextPolyhedron(P, /, n_i, c_i, f_{side})
      write DP_i, cDP_i to disk
   7. FOR each f_i \in F_i DO
      f\_list.update(f_i)
\end{verbatim}

Figure 3.15: Pseudocode of the straightforward implementation of an incremental construction method to compute the Delaunay triangulation

The procedure \texttt{Triangulate\_Inc\_Constr()} is computationally expensive and can be speeded up considerably. The two functions \texttt{FirstPolyhedron()} and \texttt{NextPolyhedron()} scan the whole dataset and compute for excessively many quadruples of input points a circumsphere. However it is intuitively clear that two points, which are far apart, generally have little or no effect on each other. This rule of thumb opens up two possible ways to reduce the amount of points that must be tested:

1. The Divide\&Conquer paradigm can be used to successively subdivide the pointset.
2. The concept of local processing allows to restrict the search space. Local processing means that to find the site \( s \) which together with an already computed Delaunay face \( f \) forms a new
3.3 Computation of the Delaunay Triangulation

Delaunay polyhedron the sites should be considered in order of increasing distance from the face $f$. Furthermore the fact can be used that only points lying inside the smallest (with respect to the signed radii) of all the previously computed circumspheres can eventually produce a polyhedron with a circumsphere having an even smaller signed radius.

We will first describe how the procedure `Triangulate_IncConstr()` can be extended to become a D&C method and then incorporate the local processing concept.

The general structure of a Delaunay triangulation procedure using the D&C paradigm is as follows:

```
procedure Triangulate_D&C(P)
1. IF (size(P) < min.size) THEN
2. DT(P) = Triangulate_IncConstr(P)
ELSE
3. h = <next partition plane>  
   {P+,P~} = Partition_Pointset(P,h)
4. DT+(P) = Triangulate_D&C(P+)
   DT-(P) = Triangulate_D&C(P-)
5. DT(P) = Merge(DT+(P), DT-(P))
6. RETURN DT(P)
```

The problem is recursively broken up into ever smaller pieces. When it is small enough (step 1) a conventional Delaunay triangulator can be used in step 2 to solve the problem. Otherwise the pointset is split into two more or less equal sized sets (step 3) and the D&C triangulator used to compute the Delaunay triangulation of the two subsets (step 4). Finally the two results must be merged in step 5. The difficulty of the D&C method lies in the merge phase. The merging requires a number of local modifications on both Delaunay triangulations which is an expensive task. As we have seen in subsection 3.2.4 there exists only in 2D an efficient solution to this problem. However Cignoni et al. [42] have proposed a relatively simple way out of the problem. Instead of merging partial results in a second step, the merging part of the Delaunay triangulation is built first and then the Delaunay triangulations of the two subsets of input points are computed independently from each other while the merging part is given as a boundary condition.

As an example let us illustrate this method on an artificial object composed of a cuboid and an elongated cylinder. The surface has been approximated by 3840 boundary points. The first partitioning plane (orthogonal to the x-axis) subdivides them into two sets containing 1808 and 2032 points respectively. The merging part of the triangulation at this level is shown in the upper left image of figure 3.16. Then the halfspace to the right of the plane is further subdivided, this time by a plane orthogonal to the y-axis. This plane partitions the 1808 points into two new subsets, one containing 850 points and the other 958. Then the new merging part of the triangulation is shown in the upper right image of the figure. In the lower row the next two recursions are shown.

The pseudocode of this triangulator is given in listing 3.17. The procedure has now as an additional argument the list `f_list` which holds the faces of the polyhedra that have already been computed in the merging phase of the triangulator of the next higher level and contain only sites from the actual pointset. This increases efficiency since no faces that once have been computed are lost. In step 3 a partition plane must be chosen. We propose to select it parallel to one of the coordinate axes alternating between the three axes. Furthermore it is positioned so that no input site is lying on it. The partitioning can then be done very efficiently since only one coordinate of the points has to be compared with the same coordinate of the plane. Then the already known faces are distributed among three face lists. `f_list` contains the faces which have only vertices in $P^-$ and `f_list` those which have only vertices in $P^+$. The third list `f_list` contains the faces which have vertices both in $P^-$ and in $P^+$, i.e. the faces that are intersected by
3.3 Computation of the Delaunay Triangulation

Figure 3.16: The first four recursions of the D&C Delaunay triangulator

the partitioning plane $h$. In step 5 those polyhedra of the triangulation are computed which are intersected by the partitioning plane $h$. This is the merging part of the Delaunay triangulation. This part of the triangulation will also contain faces which are not intersected by the plane $h$. These faces are added to the proper face list in step 6. Then the subsets $P^-$ and $P^+$ are triangulated in a recursive call of the procedure Triangulate_Reversed_D&C() (step 7). Finally the result is the union of the three partial triangulations.

The procedure Triangulate_Intersected() (see listing 3.18) is a slight variation of Triangulate_IncConstr(). If the face list handed over to the procedure is not empty there is no need to compute a first polyhedron from scratch and this part of the procedure is skipped in step 1. Otherwise the function FirstPolyhedron() has to return a polyhedron which is intersected by the plane $h$. To comply with this condition the function is adapted in the following way: It selects as the first vertex the site nearest to the plane $h$ and as the second vertex the nearest one of the vertices lying on the other side of $h$. In steps 4 and 9 we store now only those faces the face list which are intersected by the plane $h$ since the others do not necessarily generate a second polyhedron which is intersected by the plane $h$. When no plane is given all
3.3 Computation of the Delaunay Triangulation

procedure Triangulate.Reversed_D&CD(P, f_list)

1. IF (size(P) < min.size) THEN
   2. $DT^0(P) = \text{Triangulate.Integrated}(P, 0, f_list)$
   ELSE
   3. $h = \langle$next partition plane$angle$
      $\{P^+, P^-\} = \text{Partition.Pointset}(P, h)$
   4. $f_{list}^- = \{f_i \in f_{list} | f_i \cap H^+_h \neq \emptyset\}$
      $f_{list}^0 = \{f_i \in f_{list} | f_i \cap h \neq \emptyset\}$
      $f_{list}^+ = \{f_i \in f_{list} | f_i \cap H^-_h \neq \emptyset\}$
   5. $DT^0(P) = \text{Triangulate.Integrated}(P, h, f_{list}^0)$
   6. $f_{list}^- = f_{list}^- \cup \{f_i \in DT^0(P) | f_i \cap H^+_h \neq \emptyset\}$
   7. $DT^0(P) = \text{Triangulate.Reversed.D&CD}(P^+, f_{list}^+)$
   8. $DT^0(P) = \text{Triangulate.Reversed.D&CD}(P^-, f_{list}^+)$
   9. RETURN $DT^-(P) \cup DT^0(P) \cup DT^+(P)$

Figure 3.17: Pseudocode of a D&CD Delaunay triangulator which anticipates the merge phase

procedure Triangulate.Integrated(P, h, f_list)

1. IF (f_list$^0 == \emptyset$) THEN
   2. $D_{P_0} = \text{FirstPolyhedron}(P, h)$
      write $D_{P_0}, c_{D_{P_0}}$ to disk
   3. $F_0 = \text{hull}(D_{P_0})$
   4. FOR each $f_i \in F_0$ DO
      IF ($f_i \cap h \neq \emptyset$) OR ($h = 0$) THEN
      f_list$^0$.update($f_i$)
   5. WHILE (size(f_list$^0$) > 0) DO
   6. $f = f_{list}^0$.next()
   7. $D_{P_i} = \text{NextPolyhedron}(P, f, \overline{f_i}, c_f, f_{side})$
      write $D_{P_i}, c_{D_{P_i}}$ to disk
   8. $F_i = \text{hull}(D_{P_i})$
   9. FOR each $f_i \in F_i$ DO
      IF ($f_i \cap h \neq \emptyset$) OR ($h = 0$) THEN
      f_list$^0$.update($f_i$)

Figure 3.18: Pseudocode of the procedure which computes the part of a Delaunay triangulation inter¬sected by a plane $h$

faces are inserted in the face list and the complete triangulation is computed. This property is used in the procedure Triangulate.Reversed_D&CD() to stop the recursion when the pointsets become small enough.

Let us now come to the concept of local processing. The data structure that supports local processing is the uniform grid. It has been invented in 1978 by Franklin [64]. The uniform grid partitions the Euclidean space into rectangular cells, in the following called buckets. The buckets, we use, are all cubes of the same size. This choice makes the operating with the uniform grid very easy and therefore fast. By means of the uniform grid we can restrict a search to a
subpart of the Euclidean space. It is sufficient to retrieve the buckets within a specified subpart of space and to test the points contained in these buckets. Hence the search becomes local. The uniform grid can be seen as a fast indexing scheme of the data the design of which has to cover two phases:

1. The establishing of the grid and the insertion of the data into it in a preprocessing phase.

2. The fast retrieval of the data during an operating phase.

To establish the grid we first determine the integer bounding box of the data. Assume that we have \( n \) input points \( p_i = (x_i, y_i, z_i), i = 1, \ldots, n \). Then the bounding box of the data is

\[
\begin{align*}
x_{\min} &= \min_{i=1, \ldots, n} (x_i), & x_{\max} &= \max_{i=1, \ldots, n} (x_i) \\
y_{\min} &= \min_{i=1, \ldots, n} (y_i), & y_{\max} &= \max_{i=1, \ldots, n} (y_i) \\
z_{\min} &= \min_{i=1, \ldots, n} (z_i), & z_{\max} &= \max_{i=1, \ldots, n} (z_i)
\end{align*}
\]

From the bounding box we determine the length \( l \) of the buckets such that it is an integer value near the exact value for which there is in the average one bucket per input point:

\[
l = \left\lfloor \frac{(x_{\max} - x_{\min}) \cdot (y_{\max} - y_{\min}) \cdot (z_{\max} - z_{\min})}{n} \right\rfloor
\]

From the bucket length and the bounding box we finally compute the number of buckets in each coordinate direction. Thereby we have to pay attention to the special case of points lying on the grid planes since all points must be assigned to exactly one bucket. We have chosen to put the points lying on the grid planes always into the bucket to the right of the plane. That’s why to the right of the bounding box one more bucket is added and the number of buckets becomes:

\[
m = m_x \cdot m_y \cdot m_z
\]

\[
m_x = \left\lfloor \frac{x_{\max} - x_{\min}}{l} \right\rfloor + 1
\]

\[
m_y = \left\lfloor \frac{y_{\max} - y_{\min}}{l} \right\rfloor + 1
\]

\[
m_z = \left\lfloor \frac{z_{\max} - z_{\min}}{l} \right\rfloor + 1
\]

Having chosen the bucket size and the bounding box size to be integer values the determination to which bucket an input point belongs becomes very efficient. The three bucket indices \( b_x, b_y \) and \( b_z \) for a point \( (x_i, y_i, z_i) \) are then

\[
b_x = \left\lfloor \frac{x_i - x_{\min}}{l} \right\rfloor
\]

\[
b_y = \left\lfloor \frac{y_i - y_{\min}}{l} \right\rfloor
\]

\[
b_z = \left\lfloor \frac{z_i - z_{\min}}{l} \right\rfloor
\]

The uniform grid is represented as an array \( G[1 \ldots m] \) of buckets. Each bucket is a linked list of integers which are the indices into the array \( P[1 \ldots n] \) of input points. The first element of each bucket’s list does not represent an index but the number of points in this bucket. A second
array $S[1...m]$ of the same size as the grid is used to represent the state of the buckets. This gives us the possibility to search points in a stepwise growing region of space.

In the operating phase we must be able to retrieve the data efficiently. To optimally support the Delaunay triangulation algorithms there must be an operation which retrieves all points contained in a sphere. However it is not a simple task to retrieve all buckets inside a given sphere. Therefore we adopt a simpler strategy and retrieve all buckets inside the bounding box of the sphere. This will cause our algorithms to examine a few more buckets than strictly necessary while avoiding the complicated approximation of a sphere by cubic cells. The function which determines the buckets inside the bounding box of a sphere is called $get\_unvisited\_buckets\_in\_square()$ and its pseudocode is given in listing 3.19.

```python
function get_unvisited_buckets_in_square(c, r)
    1. bucketList = {}
    2. $b_{min_x} = \max(\frac{\|c-c\|_1 - r}{f(-1)}, 0)$, $b_{max_x} = \min(\frac{\|c-c\|_1 + r}{f(0)}, m_x)$

    $b_{min_y} = \max(\frac{\|c-c\|_1 - r}{f(-1)}, 0)$, $b_{max_y} = \min(\frac{\|c-c\|_1 + r}{f(0)}, m_y)$

    $b_{min_z} = \max(\frac{\|c-c\|_1 - r}{f(-1)}, 0)$, $b_{max_z} = \min(\frac{\|c-c\|_1 + r}{f(0)}, m_z)$

    3. FOR $i \in [b_{min_x}, b_{max_x}], j \in [b_{min_y}, b_{max_y}], k \in [b_{min_z}, b_{max_z}]$ DO
    4. IF ($S[i,j,k] == \text{not visited}$) THEN
        bucketList = bucketList $\cup$ $G[i,j,k]$
        $S[i,j,k] = \text{visited}$
    5. RETURN bucketList
```

Figure 3.19: Pseudocode of the function retrieving the buckets inside the bounding box of a sphere

In step 2 the bucket indices of the buckets lying inside the space covered by the uniform grid and inside the bounding box of the sphere centered at $c$ with radius $r$ are determined. In step 4 the buckets which have not yet been visited are retrieved. Of course there must be another function which resets the state of all buckets to initiate a new search from scratch. This function is called $reset\_grid()$.

As we have seen a new Delaunay polyhedron $DP_i = \{v_0, ..., v_k\}$ is constructed from a Delaunay face $f = \{v_0, v_1, v_2\}$ by searching the missing points in one of the two halfspaces $H_f^{+v_3}$, $H_f^{-v_3}$ of the face. Each point we find during a scan of the appropriate halfspace forms a polyhedron with the face $f$. This polyhedron has a circumsphere with center $c$ and signed radius $r$. Then we know that all points lying outside this sphere form a polyhedron whose circumsphere has a bigger signed radius. Hence it is pointless to scan the space outside the previously found spheres. This observation leads to an optimal scanning strategy of the uniform grid.

1. begin with the buckets inside the appropriate halfspace, let's say $H_f^{+v_3}$ and contained in the bounding box of the sphere $(c_0, r_0)$ centered at $c_0 = c_f$ (the center of the face's circumcircle) with radius $r_0 = c_0v_0$, the radius of the face's circumcircle.
2. if no point has been found move the center by a small amount $\delta$ along the line $c_f + \alpha n_f$. This gives a new sphere $(c_i, r_i)$ with center $c_i = c_0 + i * \delta * n_f$ and radius $r_i = c_i v_0$. So next search the buckets inside $H_f^{-v_3}$ and contained in the bounding box of this sphere.
3. Repeat step 2 until a non-empty sphere $(c_k, r_k)$ is found.
4. Inside the intersection of halfspace $H_f^{-v_3}$ and the bounding box of this sphere find the point $p_j$ which produces a polyhedron with a circumsphere $(c_{p_j}, r_{p_j})$ having minimal signed radius
3.3 Computation of the Delaunay Triangulation

It is possible that \( p_j \) has been found outside the sphere \((c_k, r_k)\) but inside its bounding box. That’s why the buckets inside intersection of halfspace \( H_{f}^{v_3} \) and the bounding box of the circumsphere \((c_p, r_p)\) must be searched, too.

To apply this scanning strategy the uniform grid must offer another function called `get.unvisited.buckets_in_square2()` which returns the buckets inside a sphere’s bounding box and on a given side of a specified face. It’s pseudocode is given in listing 3.20. The function

```python
function get.unvisited.buckets_in_square2(c, r, f = \{v_0, v_1, v_2\})
1. bucket_list = {}
2. REPEAT
   change the indices x, y, z and i, j, k cyclically
UNTIL (n_{f} \neq 0)
3. b_{min_x} = max(\frac{\|c_x - r\| - x_{min}}{l}, 0), b_{max_x} = min(\frac{\|c_x + r\| - x_{min}}{l}, m_x)
   b_{min_y} = max(\frac{\|c_y - r\| - y_{min}}{l}, 0), b_{max_y} = min(\frac{\|c_y + r\| - y_{min}}{l}, m_y)
   b_{min_z} = max(\frac{\|c_z - r\| - z_{min}}{l}, 0), b_{max_z} = min(\frac{\|c_z + r\| - z_{min}}{l}, m_z)
4. \( v_0' = v_0 + \text{sgn}(f_{side}) \times \frac{\|c_i\|}{\|c_j\|} \times \sqrt{3}l \)
   \( v_1' = v_1 + \text{sgn}(f_{side}) \times \frac{\|c_i\|}{\|c_j\|} \times \sqrt{3}l \)
5. \( p_y = \left( \frac{n_{x} v_0' - n_{y} v_1'}{n_{z}}, \frac{n_{y} v_0' - n_{z} v_1'}{n_{x}}, 1 \right) \)
6. FOR i \in [b_{min_x}, b_{max_x}], j \in [b_{min}, b_{max}] DO
7.   IF (n_{f} \neq 0) THEN
   8.     y = \frac{n_{x} v_0' - n_{y} v_1'}{n_{z}}
     b_y = \frac{ly}{l}
9.   ELSE
10.    IF (p_y \in H_{f}^{v_3}) THEN
11.       FOR j \in [b_{min_y}, b_{max_y}] DO
12.          IF (S[i, j, k] == not visited) THEN
13.              bucket_list = bucket_list U G[i, j, k]
14.              S[i, j, k] = visited
15.          ELSE
16.          END IF
17.       ELSE
18.          IF (p_y \in H_{f}^{v_3}) THEN
19.              FOR j \in [b_{min_y}, b_{max_y}] DO
20.                  IF (S[i, j, k] == not visited) THEN
21.                      bucket_list = bucket_list U G[i, j, k]
22.                      S[i, j, k] = visited
23.                  ELSE
24.                  END IF
25.              ELSE
26.              END IF
27.          END IF
28.       END IF
29.     END IF
30.   ELSE
31.   END IF
32. END FOR
33. RETURN bucket_list
```

Figure 3.20: Pseudocode of the function retrieving the buckets lying both inside the bounding box of a sphere and on one side of a face.
3.3 Computation of the Delaunay Triangulation

is quite complex and merits a careful explanation step by step.

**line 2** The face's normal must have at least one non-zero component. We change the indices so that we can refer to this component by the z-coordinate. This step is only a question of notation and length of the pseudocode.

**line 3** The bounding box of the sphere is determined.

**line 4** The face \( f \) is translated by \( \sqrt{3}l \) and becomes \( f' \) so that all buckets that we want to retrieve (particularly those intersected by the plane \( h_f \)) are contained in the halfspace \( H_{f'}^{v_3} \).

**line 5** \( p_y \) is a point on the y-axis which comes before the intersection point of plane \( h_f \) with the y-axis when scanning in the positive y-direction.

**lines 6-10** for each pair \((i_0, k_0)\) of bucket indices we determine the minimal and the maximal index \( j_{\min} \) and \( j_{\max} \) along the y-axis of the buckets in the halfspace \( H_{f'}^{v_3} \) and inside the bounding box of the sphere.

**line 7** We have to distinguish two cases:

(a) If the face's normal is zero in the y-direction, either all buckets in the bounding box or no one must be taken. (cf. line 10)

(b) Otherwise only the buckets before or after the plane \( h_f \) (according to where the halfspace \( H_{f'}^{v_3} \) is) must be taken.

**line 8** In case (b) the intersection point \( y \) of the straight line defined by the lower left edges of the buckets with indices \( i = i_0 \) and \( k = k_0 \) and the plane \( h_f \) is determined and the index \( b_y \) of it's bucket is computed.

**line 9** According to whether the point \( p_y \) before the plane \( h_f \) lies in halfspace \( H_{f'}^{v_3} \) or not the buckets with indices \( i = i_0, k = k_0 \) and \( j = [b_{\min}, b_y] \) or \( j = [b_y, b_{\max}] \) are retrieved.

Now we are ready to give an efficient version of the function \( \text{NextPolyhedron()} \) in listing 3.21. It implements the optimal scanning strategy described above.
function NextPolyhedron(P, f = \{v_0, v_1, v_2, \ldots, v_k\}, n_f, c_{DP_i}, side)

reset_grid()

r_{\text{max}} = \text{spatial diagonal of bounding box of input points}
\alpha = n_f \cdot (c_{DP_i} - v_0)
\delta_f = c_{DP_i} - \alpha \cdot n_f

bucket_list = get_unvisited.buckets.in.square2(c_f, r_f, f)

i = 0

WHILE empty(bucket_list) DO

i = i + 1

\hat{c}_i = \delta_f \cdot i
\hat{r}_i = \hat{c}_i \cdot v_0

IF (\hat{r}_i > r_{\text{max}}) RETURN \{0\}

bucket_list = get.unvisited.buckets.in.square2(c_i, r_i, f)

1. FOR each \(p_i \in \text{bucket_list} \) DO

\(c_{p_i} = \text{circumsphere}(f, n_f, c_f, p_i)\)
\(r_{p_i} = c_{p_i} \cdot v_0\)

4. \(v_{i3} = p_i, \ min_{p_i \in \text{bucket_list}} (r_{p_i})\)

6. cospherical.vrts = \(\{p_i \neq v_0, v_1, v_2, v_{i3}, \ldots, v_k \mid \delta(p_i, c_v) = \delta(v_0, c_v) + \epsilon\}\)

bucket_list = get.unvisited.buckets.in.square2(c_{v_{i3}}, \|r_{v_{i3}}\|, f)

1. FOR each \(p_i \in \text{bucket_list} \) DO

\(c_{p_i} = \text{circumsphere}(f, n_f, c_f, p_i)\)
\(r_{p_i} = c_{p_i} \cdot v_0\)

IF (\(r_{p_i} < r_{v_{i3}}\)) THEN
\(v_{i3} = p_i\)

cospherical.vrts = \{\}

IF (\(r_{p_i} = r_{v_{i3}}\)) THEN

cospherical.vrts = cospherical.vrts + \{p_i\}

RETURN \{4+cospherical.vrts, v_0, v_1, v_2, v_{i3}, \ldots, v_k, cospherical.vrts\}

Figure 3.21: The function NextPolyhedron() using local processing
3.4 Computation of the Voronoi Diagram

Thanks to the duality relationship the Voronoi diagram can be derived from the previously computed Delaunay triangulation. For each Delaunay face $f^D$ we have stored to disk also the two references $i_0$ and $i_1$ to the Voronoi vertices $v^V_{i_0}$ and $v^V_{i_1}$ which are dual to the Delaunay polyhedra $D_{i_0}$ and $D_{i_1}$ having $f^D$ in common. This information gives us directly a Voronoi edge $e^V = v^V_{i_0}v^V_{i_1}$. The only exception to this rule are the Delaunay faces on the convex hull of the input point set since for them only one Voronoi vertex $v^V_{i_0}$ exists and their dual counterpart is a ray beginning at $v^V_{i_0}$ and going to infinity. Since there is a high probability that these unbounded Voronoi edges (and also the Voronoi faces) would be deleted again during the regularization of the Voronoi diagram we leave them away. Once the Voronoi edges have been computed we determine the Voronoi faces as the dual counterpart to the Delaunay edges in the following way:

1. for each Delaunay edge $e^D_j$ determine the Delaunay polyhedra $D_{i_0},\ldots,D_{i_k}$ to which it belongs

2. the Voronoi face is then the convex hull of the Voronoi vertices $v^V_{i_0},\ldots,v^V_{i_k}$ dual to the Delaunay polyhedra $D_{i_0},\ldots,D_{i_k}$. Note that by this procedure the unbounded Voronoi faces will be reduced to a bounded subpart of the face thereby producing an additional edge which is not a true Voronoi edge or completly eliminated. The situation is depicted in figure 3.22

![Figure 3.22: An unbounded Voronoi face and the extracted bounded subpart delimited by the dotted line](image)

The first step would entail a repeated scanning of all Delaunay polyhedra. This is a very time-consuming process. An efficient implementation uses again the concept of local processing. The Delaunay polyhedra are scanned only once. During the scan a uniform grid is established. For all Delaunay edges of each Delaunay polyhedron two entries containing a reference to the Delaunay polyhedron are generated and inserted into the uniform grid. One is inserted into the bucket where the first vertex of the edge lies and the other into the bucket of the second vertex. To find all Delaunay polyhedra to a certain Delaunay edge we have to check only the entries in the bucket which contains one of the two vertices of the edge. The Voronoi faces are then stored in a file containing for each face the references $i_0,\ldots,i_k$ to the Voronoi vertices and also the two references $j_0,j_1$ of the Delaunay vertices $v^D_{i_0},v^D_{i_1}$ of the Delaunay edge $e^D_j$.

As we don't use the Voronoi polyhedra we actually do not determine them. However it would be a simple task. One just has to collect all Voronoi faces which have a reference to the same Delaunay vertex $v^D_{i_j}$ in common.
Chapter 4

Regularization

The term regularization stands simultaneously for

- The distinction between parts of the Voronoi diagram which represent the approximate skeleton of the original continuous shape and parts of it which are due to the discretization of the shape.

- The distinction between parts of the shape's skeleton which represent large-scale features of the shape and parts of it which are due to small-scale features, e.g. geometrical disturbances on its boundary.

Two remarks have to be made right away. First that after the discretization of the original continuous shape and the extraction of the boundary points we cannot anymore distinguish between the above two types of spurious parts of the Voronoi diagram. Hence the regularization can be regarded as a coherent framework to handle both types simultaneously. Second both types of spurious elements are basically due to small geometric disturbances and hence represent the part of the Voronoi diagram which is unstable. That's why the regularization is sometimes said to 'stabilize' the skeleton. We can put this in another way and say that the parts of the Voronoi diagram which are generated by adjacent or nearby boundary points are likely to be unstable because more sensitive to slight changes in the position of the boundary points. Note that only geometric stability is intended here. The regularization cannot do anything to increase the topological stability of the skeleton as this is a problem inherent to the definition of skeletons.

Hence the task of the regularization is to extract the Voronoi skeleton from the Voronoi diagram as its stable part. At the same time it is crucial that the Voronoi skeleton is homotopically equivalent to the Voronoi diagram. To accomplish this the regularization needs

- a measure expressing the significance of a part of the Voronoi diagram. We call this a
  importance measure.

- a deletion sequence maintaining homotopical equivalence between the original and the regularized Voronoi diagram.

Since the Delaunay triangulation is dual to the Voronoi diagram we can alternatively define the regularization on this representation. Some concepts are easier to understand on the Voronoi diagram others on the Delaunay triangulation. So we will use both of them and need also an importance measure expressing the significance of a part of the Delaunay triangulation and the definition of a deletion sequence on the Delaunay triangulation.

The regularization can be represented by the flow chart in figure 4.1. Three main phases can be individuated:

- the generation of an order on the Delaunay triangulation and on the Voronoi diagram which determines the processing order.
the calculation of importance measures for elements of the Delaunay triangulation or of the Voronoi diagram. Actually we compute the importance measures for Delaunay polyhedra and Voronoi faces.

- the homotopic deletion of elements of the Delaunay triangulation and the Voronoi diagram. Here, too, the deletion of either Delaunay polyhedra or Voronoi faces is considered. Note that in this phase it must be ensured that always both representations are updated, i.e. the deletion of a polyhedron on the Delaunay triangulation implies the deletion of its corresponding vertex on the Voronoi diagram.

The arrows show the dependencies between the different phases and make apparent that the stepwise reduction of the Voronoi diagram and the Delaunay triangulation depends on the processing order which determines the order in which elements are checked for deletion and on the significance measures which tell where the deletion has to be stopped.

In the following we will first describe the regularization in 2D as a basis for understanding the generalization to 3D.
4.1 The regularization in 2D

The regularization in 2D is most easily understood on the Delaunay triangulation. The object can be envisaged as being composed of Delaunay polygons. The regularization process proceeds from the boundary of the object to its inside. It identifies those peripheral Delaunay polygons which are insignificant for representing the object at coarser scales and removes them together with the corresponding parts of the Voronoi diagram in order to obtain the Voronoi skeleton. Before a Delaunay polygon can be deleted one has to check whether the resulting object is homotopically equivalent to the original object. Such a topological check is trivial in 2D. A Delaunay polygon is deletable iff all but one of its edges are on the object boundary.

![Figure 4.2: 2D Voronoi diagram and Delaunay triangulation with hierarchical order of Delaunay polygons](image)

The processing order in which the polygons are checked for deletion is given by the special structure of the 2D Voronoi diagram. For a simply connected object without holes it is an acyclic graph representing a tree. Hence every Voronoi edge connects exactly two parts of the Voronoi diagram. This enforces a hierarchy between branches, defining a natural order when progressing from the outmost branches to the inside of the Voronoi diagram. At the same time an order is defined on the Delaunay polygons by duality. The situation is depicted by figure 4.2.

Among the importance measures we can distinguish two different types:

- global importance measures
- local importance measures
4.1 The regularization in 2D

Figure 4.3: Subpart definition in 2D for global significance measures

(a) (b) (c)

Figure 4.4: Three importance measures in 2D. Two of them are global ones: (a) maximal distance between original and new object boundary and (b) length difference between original and new object boundary. The other is a local one: (c) pointedness of the Delaunay triangle

Global importance measures of an element are defined on a subpart of the shape assigned to that element (see figure 4.3). In the figure the hatched part of the object is the subpart assigned to the Delaunay polygon $P_i$. In 2D we can use the fact that the Delaunay edge dual to the Voronoi edge to be characterized cuts the object in two parts. Local importance measures of an element are only influenced by those boundary points which either are taking directly part in the formation of the element or are lying in a small neighborhood of such points.

In 2D several measures have been proposed in the literature. They are illustrated in figure 4.4:

(a) Brandt and Algazi [30] use the maximal distance between the original outline segment of the object and the Delaunay edge it is replaced by.

(b) Ogniewicz [129] defines four similar measures based on the comparison between the original outline segment and the Delaunay edge it is replaced by. The simplest is the 'potential residual' which is defined as the length of the original outline being replaced. In figure 4.4 it is designated by $l_{\text{orig}}$. A second measure is the 'chord residual'. It is defined by the length difference of the original outline segment and the Delaunay edge, i.e. $l_{\text{orig}} - l_{\text{new}}$. As a third measure the 'circularity residual' is defined by substituting $l_{\text{new}}$ with the disk perimeter of the polygon's circumcircle from the startpoint to the endpoint of $l_{\text{new}}$. The forth measure is called 'bicircularity residual' and is obtained by substituting in the same way all the Delaunay edges making up $l_{\text{orig}}$. 
(c) Meyer [117], Talbot and Vincent [177] as well as Attali and Montanvert [11] propose to use the angle between the fire-fronts which meet at the selected skeleton branch. It can be easily shown, that the 'pointedness' of the appropriate Delaunay triangles provides the same measure. In figure 4.4 the two equivalent angles for an example Delaunay triangle are given. In contrast to the other measures this is a local one since for its computation only information encoded in a single triangle is needed.

The last phase is the homotopic deletion of the Delaunay polygons. This can be made in a recursive manner by processing the polygons according to their order and removing those which are deletable as defined above. The topological check used in the generation of the order is needed here again since not all processed polygons have also been deleted. In fact the consecutive deletion of polygons is stopped whenever the importance measure exceeds a user-supplied threshold. All kinds of measures (such as the previously discussed ones) can be used for this purpose. The recursive deletion ensures that no topological changes occur. Note, however, that the first two of the proposed measures change monotonically during the successive deletion procedure, therefore simple thresholding will also guarantee the preservation of homotopy. This is not the case for the 'pointedness' measure, which means that it must be combined with the recursive deletion algorithm. In this case, however, small pointed triangles on the object boundary can block the regularization process, in effect making irrelevant branches of the Voronoi diagram undeletable. This is a general problem of non-monotonic regularization measures.

In the next sections a detailed discussion of the generalization of these concepts to 3D follows. The presentation is in the same order. First the topological checks needed to determine whether a Delaunay polyhedron or a Voronoi face are deletable or not are presented. Then these checks are used to determine a processing order. Afterwards the computation of local and global importance measures is addressed.

4.2 Homotopic equivalence in 3D

Let us consider for a moment the regularization as a stepwise reduction of an initial Delaunay triangulation or Voronoi diagram by deleting single Delaunay polyhedra when operating on the DT or Voronoi faces when operating on the VD. During this process we must be able to guarantee that after each step the remaining part of the DT is homotopically equivalent to the initial DT and in the same way that the remaining part of the VD is homotopically equivalent to the initial VD. The Delaunay triangulation of a set of points has been described as a cell complex \( D \) and the Voronoi diagram as a cell complex \( V \). To determine the Voronoi skeleton of the object represented by the set of points we have to consider only a subcomplex \( V_0 \subseteq V \) which represents the 'inner' Voronoi diagram of the object. \( V_0 \) is the cell complex consisting only of those elements which are strictly lying inside the object. This choice of the initial VD has the advantage that we don't have to represent unbounded Voronoi elements. Such elements exist e.g. between two neighboring points of the convex hull of the point set. A second advantage is that we don't have to represent 3-dimensional Voronoi elements since all of them contain in their interior a boundary point and hence must intersect the boundary of the object. By duality a subcomplex \( D_0 \subseteq D \) is defined on which the regularization will take place. \( D_0 \) is an open cell complex. Conversely \( V_0 \) is a closed subcomplex.

The regularization changes the initial Delaunay triangulation represented by the cell complex \( D_0 \) gradually by removing one Delaunay polyhedron after the other. Thereby a sequence of subcomplexes representing the reduced triangulations \( (D_i) \) is produced. Formally

\[
D_{i+1} = D_i \setminus S_i
\]

The subcomplex \( S_i \) which is removed at the \( i \)-th step must be a closed cell complex so that \( D_{i+1} \) is again an open subcomplex. Hence if we want to remove a single polyhedron \( p_i \) in the \( i \)-th step we must also remove the elements which bound \( p_i \). So the \( i+1 \)-th cell complex can be described as

\[
D_{i+1} = D_i \setminus \{p_i, \partial(p_i)\}
\]
4.2 Homotopic equivalence in 3D

\( \mathcal{D}_i \) and \( \mathcal{D}_{i+1} \) are homotopically equivalent if their boundaries are homotopically equivalent. Hence we must ensure that at each step the boundary of the newly generated subcomplex is homotopically equivalent (denoted by \( \cong \)) to the previous one: \( \partial(\mathcal{D}_i) \cong \partial(\mathcal{D}_{i+1}) \). The deletion of \( p_i \) changes the boundary of \( \mathcal{D}_i \) only locally, namely at the bounding elements of \( p_i \). This means that the homotopical equivalence of \( \mathcal{D}_i \) and \( \mathcal{D}_{i+1} \) can be decided locally by a deletability criterion based on the membership of the bounding elements of \( p_i \) to \( \mathcal{D}_i \) or to \( \mathcal{D}_{i+1} \).

In the same way the regularization produces a sequence \( \{V_i\} \) of cell complexes representing the stepwise reduced \( \mathcal{V}_D \). This time the elements we want to remove at each step are the Voronoi faces \(^1\). Let us again formalize this process. From the \( i \)-th closed cell complex \( V_i \) a single 2-dimensional element (a Voronoi face) \( f_i \) is removed. There might be some of the bounding elements of \( f_i \) which bound no other elements than \( f_i \) and thus must be removed together with \( f_i \). Hence the \( i+1 \)-th cell complex can be described as

\[
V_{i+1} = V_i \setminus \{f_i \cup \{c \in \partial(f_i) | \exists c' \in V_i \setminus \partial(f_i) : c \preceq c'\}\}
\]

Again \( V_{i+1} \) and \( V_i \) must be homotopically equivalent. A deletability criterion for Voronoi faces will be presented in subsection 4.2.2.

4.2.1 Deletability criterion for Delaunay polyhedra

Whether a Delaunay polyhedron is deletable or not can only be determined relative to the current subcomplex \( \mathcal{D}_i \). On the Delaunay polyhedra contained in \( \mathcal{D}_i \) we define a binary decision function \( \kappa_i \) which maps the polyhedra to the binary set \( \{0, 1\} \). The value 1 is assigned to a polyhedron \( p_k \) if the boundary of \( \mathcal{D}_i \) is homotopically equivalent to the boundary that results when \( p_k \) and the bounding elements of \( p_k \) are removed from \( \mathcal{D}_i \). We call this function deletability criterion.

Here is the formal definition:

**Definition 8**

\[
\kappa_i(p_k) = \begin{cases} 
1 & \text{if } \partial(D_{i+1}) \cong \partial(D_i), D_{i+1} = D_i \setminus \{p_k, \partial(p_k)\} \\
0 & \text{otherwise}
\end{cases}
\]

**Figure 4.5:** A single Delaunay polyhedron of a Delaunay triangulation. The rest of the triangulation is not shown. The bounding elements of the polyhedron have been shaded according to whether they belong to the boundary of the triangulation (no shading) or they are cutting through the triangulation (grey shaded).

The deletability criterion is defined locally on the bounding elements of \( p_k \). They can be grouped into two categories according to whether they belong to the boundary of the current subcomplex \( \mathcal{D}_i \) or whether they are cutting through it. The elements which belong to the boundary are called outside elements, the others inside elements. In figure 4.5 a polyhedron has been cut out from some cell complex as an example. The inside faces are shaded in light grey and the inside edges are the dashed lines. There are no inside vertices (in our case this is generally

---

\(^1\)At the same time one must consider the loci where the Voronoi diagram has been reduced to edges, i.e. there are no Voronoi faces to remove. This poses no major problem: edges can only be removed when they have an endpoint which bounds no other element.
true since the vertices have been chosen from the boundary of the initial cell complex). The outside faces are shaded in dark grey, the outside edges are the solid lines and all the vertices are outside vertices. In order to have a complete picture of the situation a plane model of the polyhedron is given on the right hand side (the boundary of the circle corresponds to a single point of an inside face). The elements can be grouped into connected components containing only inside or only outside elements using the bounding relation $B$. Then the deletability criterion is defined by two conditions:

1. there is exactly one inside and one outside component.

2. the boundary between the two components is a non self-intersecting, closed line.

The criterion is illustrated with the examples given in figure 4.6 and in figure 4.7. In the first two examples there are exactly one inside and one outside component. But in the second of these two examples the common boundary between the inside and outside component is self-intersecting at point $Q$. Consequently $p_k$ is deletable only in the first example and not in the second one, i.e. $\kappa_i(p_k) = 1$ in the first example and $\kappa_i(p_k) = 0$ in the second.

Figure 4.6: Illustration of the deletability criterion. The left polyhedron is deletable, the right one not since the boundary between the dark faces and the grey ones is self-intersecting in point $Q$.

Figure 4.7: A polyhedron which is not deletable. On the left it is shown embedded in the rest of the triangulation, on the right it is isolated.

In the third example of Figure 4.7 the polyhedron is shown as a part of the Delaunay triangulation on the left and on the right it is isolated. Again the outside elements have been
shaded. This time there are two outside and two inside regions. Hence the first condition is not fulfilled and the polyhedron is not deletable. In fact the deletion of the polyhedron would cut the Delaunay triangulation into two parts of which one would be homotopically equivalent to a torus.

Let us show now that the given criterion is sufficient to guarantee homotopical equivalence. In topology two objects are called homotopically equivalent if an invertible continuous mapping exists between them. The continuity of a mapping can be defined by the concept of neighborhood which has already been defined for cell complexes. An element \( e \) is called near a set of elements \( S \) if every neighborhood of \( e \) contains an element of \( S \). A mapping \( f : D \mapsto D' \) is called continuous if for any element \( e \in D \) and set \( S \subseteq D \) holds that if \( e \) is near \( S \) then \( f(e) \in D' \) is also near \( f(S) \subseteq D' \). In our case we have to find a continuous mapping from the elements that make up the boundary of \( D_i \) to those which make up the boundary of \( D_{i+1} \). Such a mapping can be constructed in the following way:

1. The mapping is defined to be the identity on the unchanged parts of the boundary. This includes the outside elements of \( p_k \) making up the boundary between inside and outside elements.
2. Due to the second condition of the deletability criterion the remaining outside and inside elements of \( p_k \) make up exactly two components \( P_I \) and \( P_O \), even after the removal of their common boundary. These two boundary parts are homologous because they are simply connected, thus one of them can be projected to the other.
3. The common, non self-intersecting boundary guarantees the continuity of the composition of the two homotopic mappings.

Therefore the given criterion is sufficient to guarantee homotopic equivalence. But the question remains whether it is mandatory or not.

In the implementation of the deletability criterion we only need to consider the outside components of \( p_k \) and the corresponding boundaries between inside and outside components. A component can have dimensionality 0,1 or 2.

From the fact that the Delaunay triangulation has been constructed only from points lying on the object’s boundary we directly conclude that all vertices (0-elements) are outside elements. This means that they must all belong to one single outside component. In other words if isolated vertices exist, \( p_k \) is not deletable.

The existence of 1D outside components again makes \( p_k \) not deletable as the boundary of a 1D component is not a closed line.

The boundary of 2D components consists of vertices and edges. We don’t consider the vertices since they are all outside elements. Three types of outside edges can be distinguished

1. outside edges bounding two outside faces. These are lying in the inner of the component and do not belong to the component’s boundary.
2. outside edges bounding two inside faces. These are edges where the component’s boundary is self-intersecting. Hence if such edges are found \( p_k \) is not deletable.
3. outside edges bounding both an inside and an outside face. These are the edges of the component’s boundary which together must make up exactly one closed line which does not self-intersect in a vertex. If this is true \( p_k \) is deletable.

In listing 4.8 the implementation of the deletability criterion is given in pseudocode. The function takes a polyhedron \( p_k \) as input and returns a boolean value depending on whether the polyhedron is or is not deletable. Let us step through the algorithm:

lines 1–3 Two lists of outside edges are filled. The list \( o_o \_edges \) holds the edges bounding two outside faces, the list \( o_i \_edges \) those bounding an inside and an outside face.
4.2 Homotopic equivalence in 3D

function is deletable(pk)

1. o.i.edges = make_list()
o.o.edges = make_list()
component.boundary = make_list()
2. FOR each outside face fi ∈ pk DO
   FOR each edge ei ∈ fi DO
   3. IF member(ei, o.i.edges)
      remove(ei, o.i.edges)
      insert(ei, o.o.edges)
   ELSE
      insert(ei, o.i.edges)
   4. FOR each inside face fi ∈ pk DO
      FOR each outside edge ei ∈ fi DO
      5. IF not member(ei, o.i.edges) and
         not member(ei, o.o.edges)
         RETURN not deletable
      6. IF size(o.i.edges) > 0
      7. eo = get_edge(o.i.edges)
v0 = start_vertex(eo)
v1 = end_vertex(eo)
      8. insert(v0, component.boundary)
      9. WHILE (ei=get_edge(o.i.edges, vi)) > 0 DO
      10. IF member(vi, component.boundary)
           RETURN not deletable
      11. insert(vi, component.boundary)
      12. IF start_vertex(ei) == vi
          vi = end_vertex(ei)
      ELSE
          vi = start_vertex(ei)
      13. IF size(o.i.edges) > 0
          RETURN not deletable
      14. o.vertices = make_list(component.boundary)
      15. FOR all ei ∈ o.o.edges DO
          insert(start_vertex(ei), o.vertices)
          insert(end_vertex(ei), o.vertices)
      16. IF size(o.vertices) == nr_of_vertices(pk)
          RETURN deletable
      ELSE
          RETURN not deletable

Figure 4.8: Pseudocode of the deletability criterion for Delaunay polyhedra

lines 4–5 Outside edges bounding two inside faces are detected. If such edges are found the function can immediately return with a negative answer. Notably this addresses not only the case of 1D but also 2D components with a boundary that is self-intersecting along an edge.

lines 6–12 One of possibly several component boundaries is constructed as follows. An arbitrary edge is taken from the list o.i.edges (function get_edge()), one of its vertices (v0) is added to the list component.boundary and the other (vi) used as a parameter in the function get_edge() to retrieve the next edge (ei) starting at vi. After a test whether the vertex vi
of the new edge is already part of the component’s boundary, and therefore the boundary would be self-intersecting (line 10), it is added to the list component.boundary (line 11). This is done repeatedly until the component’s boundary is complete.

**line 13** At this point the list o.i.edges should be void, otherwise there are more than one outside components and the function returns with a negative answer.

**lines 14–16** The last check is whether all vertices make part of the single outside component we found. This is done by collecting all vertices belonging to outside edges in the inner of the component and the vertices of the component’s boundary in the list o.vertices. If o.vertices contains all vertices of \( p_k \), \( p_k \) is deletable otherwise it is not.

With the function is.deletable() we can find all polyhedra of the cell complex \( D_l \) which are deletable. According to a processing order (which will be defined in section 4.3) one of these polyhedra together with its bounding elements is then deleted and the new subcomplex \( D_{l+1} \) is obtained.
4.2.2 Deletability criterion for Voronoi faces

The deletability criterion for Voronoi faces is similarly defined

**Definition 9**

\[ \kappa_i(f_k) = \begin{cases} 
1 & \text{if } (V_{i+1} \cong V_i), \quad V_{i+1} = V_i \setminus \{f_k \cup \{c \in \partial(f_k) \mid \exists c' \in V_i \setminus \partial(f_k) : c \neq c'\}\} \\
0 & \text{otherwise} 
\end{cases} \]

Hence \( f_k \) is deletable if \( V_i \) is homotopically equivalent to \( V_{i+1} \). This depends on the bounding elements of \( f_k \) which are 0-dimensional and 1-dimensional elements, i.e. vertices and edges. Again we define inside and outside elements:

- The edges which bound exactly one 2-dimensional face are outside elements, those bounding two faces are inside elements.
- Each vertex \( v_i \) bounds two edges \( e_0 \) and \( e_1 \) of the boundary of \( f_k \) and eventually additional edges which do not belong to the boundary of \( f_k \). The vertices are outside elements if one of the following conditions hold:
  1. \( e_0 \) and \( e_1 \) are both outside elements and \( v_i \) bounds no other edge.
  2. \( e_0 \) or \( e_1 \) are not both outside elements and \( v_i \) bounds at least one outside element, no matter whether that element belongs to the boundary of \( f_k \) or not.

All other vertices are inside elements.

The elements can be grouped into inside and outside components. For illustrative means some examples are given in figure 4.9. The inside elements are dark shaded. \( V_i \) and \( V_{i+1} \) are homotopically equivalent if a continuous invertible mapping between the boundary elements of the two cell complexes exist. Such a mapping can be given if:

1. there is exactly one inside and one outside component.
2. outside and inside components are both 1-dimensional. In other words none of them is degenerated to a single vertex.

![Figure 4.9: Illustration of the deletability criterion for Voronoi faces. \( f_k \) is deletable only in the first configuration.](image)

The mapping can then be easily constructed

1. The mapping is defined to be the identity on the unchanged parts of the boundary.
2. The remaining parts of the boundary belong to the outside component of \( f_k \) and must be mapped to the inside component. The second condition ensures that the two components are two simply connected lines. These lines generally are made up of a number of outside and inside edges respectively. Due to the invariance principle they can be subdivided, so that both components contain the same number of edges. Then the mapping can be defined by identification of the endpoints of these edges.
4.2 Homotopic equivalence in 3D

Figure 4.9 shows four cases. On the left a constellation is shown where $f_k$ is deletable, in the middle two situations are sketched where the deletion of $f_k$ would make $V_{i+1}$ disconnected, since there are two inside and two outside components. On the right there is an example that fulfills the first condition but not the second. In that case the deletion of $f_k$ would create a hole in $V_{i+1}$.

In the implementation of the deletability criterion we consider the outside edges of $f_k$. Only if at least one edge of $f_k$ is an inside element, a inside component with dimensionality 1 exists. Hence if all edges of $f_k$ are outside edges $f_k$ is not deletable. Otherwise a chain of outside edges connected by outside vertices is constructed starting at an arbitrary outside edge. Then $f_k$ is deletable if this chain contains all outside edges and vertices of $f_k$, otherwise more than one outside component exists and $f_k$ is not deletable.

function is_deletable($f_k$)

1. o.edges = make_list()
   o.vertices = make_list()
2. FOR each outside edge $e_i \in f_k$ DO
   insert($e_i$, o.edges)
3. FOR each outside vertex $v_i \in f_k$ DO
   insert($v_i$, o.vertices)
4. IF (size(o.edges) == 0) OR (size(o.edges) == nr_of_edges($f_k$))
   RETURN not deletable
   ELSE
5. $e_0$ = get.edge(o.edges)
   $v_{\text{head}}$ = start.vertex($e_0$)
   $v_{\text{tail}}$ = end.vertex($e_0$)
6. WHILE (member($v_{\text{head}}$, o.vertices) AND
   ($e_i$=get.edge(o.edges, $v_{\text{head}}$)) > 0) DO
   remove($v_{\text{head}}$, o.vertices)
7. IF $v_{\text{head}}$ == start.vertex($e_i$)
   $v_{\text{head}}$ = end.vertex($e_i$
   ELSE
8. $v_{\text{head}}$ = start.vertex($e_i$)
9. WHILE (member($v_{\text{tail}}$, o.vertices) AND
   ($e_i$=get.edge(o.edges, $v_{\text{tail}}$)) > 0) DO
   remove($v_{\text{tail}}$, o.vertices)
10. IF $v_{\text{tail}}$ == start.vertex($e_i$)
    $v_{\text{tail}}$ = end.vertex($e_i$
    ELSE
11. $v_{\text{tail}}$ = start.vertex($e_i$)
12. IF ((size(o.edges) == 0) AND
    (size(o.vertices) == 0))
    RETURN deletable
   ELSE
   RETURN not deletable

Figure 4.10: Pseudocode of the deletability criterion for Voronoi faces

In listing 4.10 the implementation of the deletability criterion is given in pseudocode. The function takes a face $f_k$ as input and returns a boolean value depending on whether the face is or is not deletable. Let us step through the algorithm:

lines 1 and 2 All outside edges are filled into the list o.edges. To determine whether an edge is an outside edge it is sufficient to check whether it bounds only one face of $V_i$. 

4.3 The processing order

In 2D a processing order has been defined by the special hierarchical structure of the Delaunay triangulation. It was similar to a peeling of the object. One layer of deletable Delaunay polygons after the other has been removed from the object. Polygons of the same layer were independent from each other and became the same order number. In 3D the topological structure of the cell complexes is more complicated. This leads to configurations where the deletion of a polyhedron from a layer of potentially deletable polyhedra makes another polyhedron of the same layer non-deletable. Hence in 3D the deletion of polyhedra belonging to the same layer must be made in a sequential manner. Depending on the sequence some of them will eventually become non-deletable and must be reexamined later. This yields the following algorithm:

- Establish a layer of polyhedra which are currently deletable. Put them in a queue.
- Take one polyhedron after the other from the queue, test whether it is still deletable. If it is, remove it from the queue, otherwise reinsert it at the end of the queue. All polyhedra which are removed from the queue receive the same processing order number. When no more polyhedra in the queue are deletable establish the next layer of polyhedra and add them to the queue.
- The polyhedra which remain at the end receive the highest processing order number.

A flow chart of the algorithm can be found in figure 4.11. Note that the order in which the polyhedra are put into the queue will eventually influence the outcoming processing order since the deletability of a polyhedron can change while it is in the queue. The further ahead a polyhedron is in the queue the smaller is the probability that the deletion of another polyhedron makes it become non-deletable. In practice the polyhedra are often inserted randomly and this effect is neglected.

Different processing orders can be defined by changing the way in which a new layer of polyhedra is determined. We tried two slightly different processing orders which are similar to a peeling of the object.

(a) The next layer contains all currently deletable polyhedra as determined by the deletability criterion. Since only the topological information of deletability is used we will call this processing order 'topological peeling'. In figure 4.12 some snapshots of the Delaunay triangulation and the corresponding Voronoi diagram, taken during the determination of this processing order can be seen. The first seven layer of polyhedra of the initial Delaunay triangulation on the left have been coloured according to their processing order, the polyhedra of the other layers are all shown in blue. In the flow chart of figure 4.11 path (a) corresponds to the topological peeling.
4.3 The processing order

![Flow chart of the processing order determination](image)

**Figure 4.11: Flow chart of the processing order determination**

(b) The next layer contains all currently deletable polyhedra up to a certain depth $t_{\text{depth}}$ in the object. This way we leave out from the current layer all topologically deletable polyhedra which have a radius of circumsphere greater than $t_{\text{depth}}$. When performing regularization on the Voronoi diagram the radius of circumsphere can be determined by taking half the distance between the two generating points of the Voronoi faces. We call this processing order 'geometric peeling'. In the flow chart of figure 4.11 path (b) corresponds to it.

The weakness of topological peeling can be seen on the partly reduced Voronoi diagram of the 3-dimensional letter E in figure 4.13. In the regions marked by arrows the peeling propagates
4.3 The processing order

Figure 4.12: Some snapshots of the Voronoi diagram and the corresponding Delaunay triangulation at various order levels of the processing order, called ‘topological peeling’.

Figure 4.13: In the regions marked by arrows the topological peeling propagates much slower into the object. A more isotropic processing order can be defined using the distance to the object’s boundary.

much slower into the object. The reason for this effect is that in concave regions of the object’s boundary many small Delaunay polyhedra appear. Hence a processing order like the topological peeling which depends only on the cell complex’s topology will behave anisotropically. Geometric peeling, on the other hand, is more isotropic as can be seen on the snapshots of the successively reduced Voronoi diagram of the letter E in figure 4.14.

Both processing orders will leave many polyhedra with the same order number. These are just processed randomly. Note that the processing orders have been defined on the Delaunay triangulation. They can be defined in the same way on the Voronoi diagram simply by substituting in the above description Delaunay polyhedra with Voronoi faces.

The processing order is established before the actual regularization takes place. During the actual regularization the deletability criterion is used again since the deletion of polyhedra can be blocked when the importance measure is higher than a user-supplied threshold making previously deletable polyhedra undeletable. Hence the actual regularization removes the polyhedra, ordered according to the processing order, from the object one at a time as long as two requirements are fulfilled:

- they are still deletable and not blocked by neighboring polyhedra at which the regularization has already been stopped.
4.3 The processing order

Figure 4.14: Some snapshots of the Voronoi diagram and the corresponding Delaunay triangulation at various order levels of the more isotropic processing order, called 'geometric peeling'.

- their significance (determined by an importance measure) is less than a chosen threshold.

This process is illustrated by the flow chart in figure 4.15 where the thresholding of the polyhedra according to an importance measure can be seen very well.
Figure 4.15: Flow chart of the recursive homotopic deletion of elements
4.4 Importance measures in 3D

4.4.1 Local Importance measures

In the definition of local importance measures only local parts of the Delaunay triangulation or the Voronoi diagram are involved. Generally they can be computed in a simple and efficient way which makes them so attractive. However they have the inherent problem that globally insignificant perturbations on the boundary may have a high local importance. As an example just think of highly pointed tetrahedra on the object's boundary, which will block the regularization since they can't be deleted due to their high local importance measure. Another problem arises on the parts of the Voronoi diagram which we called the connection branches (cf. subsection 1.3.3). In such parts local measures will generally be weak since no real local symmetries are existing. Nevertheless these parts are essential to guarantee the homotopy between the skeleton and the original object. This fact is the reason for which we have to use the topological checks described in section 4.2 during the regularization. Note that some of the global measures in 2D have been proven to be monotonic which allowed the use of simple thresholding during the regularization.

The radius function

A first approach to the definition of importance measures has been given by Brandt [29] who suggests that skeletal extensions which are due to perturbations on the boundary always have 'rapidly' decreasing radius values. He looks at the radius function for a Voronoi edge connecting two Voronoi vertices \( v_0 \) and \( v_1 \). A radius value \( r_0 \) and \( r_1 \) is assigned to each vertex. Then, since the radius function is Lipschitz, he states that

\[
\frac{|r_0 - r_1|}{||v_0 - v_1||} < 1
\]

This ratio approximates the slope of the radius function by which he defines the change in the radius function. Then the radius function is said to be 'rapidly' changing if the ratio for a pair of vertices is greater than a given threshold (near one) and consequently the vertex having assigned a smaller radius is removed together with the subgraph adjacent to that vertex and having monotonically decreasing radius values. The behaviour of the radius function can be well seen in figure 1.7 of section 1.3.3. This measure has only been used for trivial cases and never been further developed in the literature.

The pointedness measure

![Diagram showing pointedness measure](image)

**Figure 4.16:** The pointedness of a salient tetrahedron is defined by the angle between the two faces on the boundary of the shape. On the left the Delaunay triangulation and the Voronoi diagram of the original shape are shown and on the right of the simplified shape.

The pointedness measure proposed in 2D by Meyer [117], Talbot and Vincent [177] has been generalized to 3D by Attali [11, 9]. Attali distinguishes between two types of deletable tetrahedra: 'hat' tetrahedra and 'salient' tetrahedra. The first ones are characterized by having three and the others by having only two faces on the object's boundary. The pointedness measure is only
4.4 Importance measures in 3D

Defined on the salient tetrahedra. It measures the angle $\theta$ between the two boundary faces as one can see in figure 4.16. Thereby $v_0$ and $v_1$ are the Voronoi vertices on the opposite side of the boundary faces with respect to vertex $v$. Note that this is also the angle between the two firefronts initiated at these boundary faces and meeting at the Voronoi vertex dual to the tetrahedron. The measure is only defined for tetrahedra and it is not straightforward to generalize it to polyhedra.

We define another local measure which is also related to the angle of meeting firefronts but is more stable with respect to discretization effects.

**The ridge strength measure**

![Figure 4.17: Sketch of an ideal 1-dimensional ridge in the distance map of a 2-dimensional object](image)

According to the definition of the SAT by the centers of all maximal inscribing balls, the skeleton of an object is equivalent to the ridges of its distance map (DM). Euclidean DM's can be generated very efficiently following e.g. a sequential algorithm proposed by Danielsson in 1980 [45]. We mentioned in chapter 2 that due to this equivalence it has already been tried to extract ridges directly from DM's both for 2D images and for 3D images. The DM of an image represents the distance of an image point to its nearest boundary point. The search for 2-dimensional distance ridges (i.e. sheets) in 3-dimensional DM's is a difficult task since one has to search for sheets with unknown parameterization. Thereby procedures based on ridge tracking become very hard to implement and generally fail in nearly flat regions of the DM leading to disconnected skeletons.

The strength of the ridges in the DM, however, can well be used for the characterization of the significance of single Voronoi faces. The detection of the ridge strength is a much easier problem, as the Voronoi faces specify the exact position of the ridge to be investigated, and even define the orthogonal direction to it. Hence we regard the Voronoi faces as candidates for distance ridges and define a ridge-strength for them based on the comparison to the model of an ideal distance ridge. The ridge-strength measure is therefore a combination of the information at hand from the VD (i.e. exact location and orientation of possible skeletal faces) and the information derived from the DM (how well a candidate face represents a ridge in the DM).

To explain the implementation of the ridge-strength measure let us first consider the model of an ideal distance ridge in a 2D image. Figure 4.17 shows the DM inside the 2-dimensional object delimited by the dashed lines. The distance ridge in the middle of the object is shown as a red line. Overlaid to the image is a 3-dimensional sketch of the ridge on two planes orthogonal to the object boundary and passing through the ridge point $P$ and the boundary points $v_r$ and $v_l$ respectively. These points are the nearest boundary points of $P$. The segments from $P$ to $v_l$ and from $P$ to $v_r$ represent the minimal length $d_{\text{min}}$ from point $P$ to the object boundary and therefore are sometimes called shortest intrusion paths. Along these segments the distance map
values uniformly decrease. In fact the DM has a gradient of unit length in their direction. This is shown on the right image of Figure 4.17. The angle \( \beta \) between a shortest intrusion path and the bisector of the segment \( \overrightarrow{v_i v_r} \) which is tangential to the ridge at point \( P \) is related to the angle of intersection \( \alpha \) of the two boundary tangents at the points \( v_i \) and \( v_r \) by the equation \( \beta = 90^\circ - \alpha \). The significance of the ridge can then be defined by the angle \( \alpha \). The more the two boundary tangents are in opposite position the smaller \( \alpha \) will be.

The same considerations can be made for points on 2-dimensional ridges in a 3-dimensional DM. In this case \( \beta \) is the angle between the bisecting plane of \( v_i \) and \( v_r \) and one of the shortest intrusion paths. \( \alpha \) is the intersection angle of the two planes defined by the two shortest intrusion paths of \( v_i \) and \( v_r \). These are tangential to the object's boundary at \( v_i \) and \( v_r \) respectively.

![Figure 4.18: The graph shows in sorted order for each Voronoi face the difference of the maximum and minimum angle that can be found between the two vectors to the generating points of the Voronoi face when varying the vector's origin on the face.](image)

The effects of discretization impede a direct implementation of this ridge-strength measure. On one hand we know the boundary of the object only at a discrete set of points, on the other hand the skeleton itself is approximated piecewise by polygons (the Voronoi faces). This means that for each Voronoi face which represents a whole set of candidate ridge points the same two boundary points (the generating points of the Voronoi face) are identified as nearest points on the object's boundary. Hence we don't know the shortest intrusion paths belonging to the points on the Voronoi face. As an experiment we computed for each Voronoi face on a uniformly distributed set of points on the face the angles between the two vectors pointing to the two generating points of the face. The differences between the minimum and the maximum angle for each Voronoi face is shown by the graph in figure 4.18. It confirms that the possible shortest intrusion paths can vary considerably.

We propose an implementation which approximates the ridge-strength of a Voronoi face from the discrete DM. By using the DM the above mentioned discretization effects are smoothed out. This is only a heuristic but it will be justified by the results.

The angle \( \alpha \) can be approximated from the DM in the direction of the ridge's normal \( \vec{n} \). In figure 4.19 one can see that the angle \( \alpha \) shows up again between the ridge's normal and its shortest intrusion path. Under the assumption that the boundary is not curved \( \alpha \) can be determined by \( \cos(\alpha) = \frac{\vec{n} \cdot \vec{a}}{|\vec{n}|} \). In the proximity of the ridge point \( R \) (magnified region) the equation \( \cos(\alpha) = \frac{\vec{n}}{\alpha} \) is a good approximation for \( \alpha \) even when the boundary is curved. The right triangle \( RQP_0 \) has
Figure 4.19: Sketch of an ideal 1-dimensional ridge in a 2-dimensional distance map which shows how the angle $\alpha$ characterizing the ridge-strength can be approximated by $\gamma$ without knowing about the shortest intrusion paths

two angles of 45 degrees at $R$ and at $P_0$. Hence $y = x$ and therefore

$$\alpha = \arccos\left(\frac{y}{x'}\right)$$

In the triangle $RQP_1$, which is perpendicular to the ridge, the angle $\gamma$ at $R$ is related to $\alpha$ by

$$\gamma = \arctan\left(\frac{1}{\cos(\alpha)}\right)$$

So $\gamma$ can be used to measure the ridge’s strength. $\alpha$ is in the interval $[0^\circ, 90^\circ]$ where its cosine is in the interval $[1, 0]$. Hence $\gamma$ lies in the interval $[45^\circ, 90^\circ]$. An angle near $45^\circ$ represents a strong ridge and an angle near $90^\circ$ a weak ridge. The angle $\gamma$ can be approximated from the discrete DM in the following way:

- The ridge is approximated at the points $r_i$, $i = 1, \ldots, n$.
- For each ridge point $r_i$ the discrete distance map $DM(x, y)$ is evaluated at some uniformly distributed sample points along the ridge’s normal $\vec{n}$ in positive and in negative direction

$$p_j = (x(j, r_i), y(j, r_i)) = (x_{r_i}, y_{r_i}) + j \cdot \vec{n}, \ j = \pm 1, \ldots, \pm m$$

This yields for every ridge point $r_i$ two sets $P_-$ and $P_+$ of points in the 3-dimensional DM.
4.4 Importance measures in 3D

Figure 4.20: The left graph shows for a Voronoi face representing a distance ridge a superposition (after a correction shift) of the distance points sampled in the distance map along lines perpendicular to the face. The graph on the right shows the derived median distance point sets.

- To compare the $2i$ point sets they must be shifted with respect to the distance value at their corresponding ridge point $r_i$ by $-DM(r_i)$.
- Then a median distance value can be computed from all distance samples at the same index $j$. Thereby two median point sets are obtained, one containing the median values at positive indices $j$, the other at negative indices $j$.
- Each of the two median point sets is approximated by a simple line fit. This gives two intersecting lines with an angle of intersection of $2 \cdot \gamma$.

The description of the computation of angle $\gamma$ has been given in 2D for the sake of simplicity. The generalization to 3D is straight forward. Instead of a line the ridge is now a face and the distance map is sampled along a line orthogonal to the ridge for each ridge point of a set of points uniformly distributed over the face. The rest of the computation procedure is identical.

The left graph of figure 4.20 shows a superposition (after the correction shift) of the distance points determined for each of the $2i$ sampling point sets. The graph on the right shows the derived median distance point sets.

The ridge strength measure is very effective in distinguishing the Voronoi faces which represent ridges in the distance map from those caused by small geometric disturbances on the boundary due to discretization effects. The reason is that perturbations on the boundary are more and more smoothed out in the distance map the further one gets away from the boundary. Hence the ridge strength measure combines in an illustrative way the ridge's distance to the object's boundary with the angle between its generating firefronts. Note that this result corresponds well to the method proposed by Attali [9] who 'stabilizes' the pointedness measure by combining it with the radius of the Delaunay polyhedra's maximal balls. The effectiveness of the ridge-strength measure is well illustrated in figure 4.21 on an artificial object which consists of a cylinder merged with a box. In the upper row the box is aligned with the image raster. One can well see the Voronoi faces due to the discretization of the curved boundary of the cylindrical part. In the lower row the same object has been slightly rotated with respect to the image raster which intensified the discretization effects. Both times the Voronoi diagram in the middle two images has been coloured according to the ridge strength measure. As one can see on the right two images the skeleton of the object can be perfectly extracted.
4.4 Importance measures in 3D

Figure 4.21: The ridge-strength measure can be used to filter out the many Voronoi faces due to discretization effects. The skeleton of the surface of a cylinder merged with a box, once aligned with the image raster and once in an oblique position, has been extracted from the Voronoi diagram.

Concluding we can say that the ridge strength measure is able to eliminate most Voronoi faces which are due to discretization effects and do not belong to the object’s skeleton. However it can not be used to express the importance of the remaining Voronoi faces with respect to the object’s shape. This is a consequence from the fact that only local parts of the object’s boundary are involved in the generation of the distance ridge corresponding to a specific Voronoi face. In order to measure the shape prominence of a Voronoi face one has to determine the whole part of the shape which is depending on it. Hence global measures are desired.

4.4.2 Global importance measures

We already mentioned that a global importance measure is based on the assignment of a subpart of the object to each element of the Voronoi diagram or the Delaunay triangulation. In the following we will call such an assignment subpart mapping.

The simple topological structure of the Delaunay triangulation in 2D allows an easy and natural definition of a subpart mapping for both the Delaunay edges and the Voronoi edges. The mapping for the Delaunay edges relies on the fact that an internal Delaunay edge always cuts the object into two disjoint components, for the Voronoi edges the crucial property is that every Voronoi edge connects two distinct parts of the Voronoi diagram, since the Voronoi diagram contains no cycles. Note that both properties only hold if the object is topologically equivalent to a point. We will make a careful analysis of the subpart mapping in 2D below.

In 3D the topological structure of both the Voronoi diagram and the Delaunay triangulation is much more complex. This can already be seen from the fact that a Delaunay face cuts the object only in insignificant degenerate cases into two parts. In order to give a clear view of the situation both in 2D and 3D it is useful to formally define at least for one element type (Delaunay edges in 2D and Delaunay faces in 3D) what a subpart mapping is. For that purpose we first define a
graph representing the elements involved in the subpart mapping.

The D-graph

The Delaunay triangulation has been defined in section 3.1 as a cell complex $\mathcal{D}$

$$\mathcal{D}(D, B_D, dim_D)$$

For the subpart mapping only a part of the basic elements will play a role, namely the Delaunay edges and polygons in 2D and the Delaunay faces and polyhedra in 3D. That's why we define the two subcomplexes $\mathcal{D}_2$ and $\mathcal{D}_3$ containing only these elements

$$\mathcal{D}_2(D_2, B_{D_2}, dim_{D_2}) \overset{def}{=} \{ d \in D_2 \iff dim_D(d) \geq 1 \}$$

$$B_{D_2} = B_D \cap (D_2 \times D_2)$$

$$dim_{D_2} = dim_D | D_2$$

$$\mathcal{D}_3(D_3, B_{D_3}, dim_{D_3}) \overset{def}{=} \{ d \in D_3 \iff dim_D(d) \geq 2 \}$$

$$B_{D_3} = B_D \cap (D_3 \times D_3)$$

$$dim_{D_3} = dim_D | D_3$$

The two complexes contain only two different types of elements and can therefore be interpreted as graphs $G = (N, E)$ where $N$ is a set of nodes and $E$ a set of edges between the nodes of $N$. In 2D the nodes are the Delaunay polygons and the edges the Delaunay edges. In 3D the nodes are the Delaunay polyhedra and the edges the Delaunay faces. Formally

$$G_2(N_2, E_2) \overset{def}{=} \{ n \in N_2 \iff (n \in D_2) \land (dim_{D_2}(n) = 2) \}$$

$$(n, n') \in E_2 \iff (n, n' \in N_2) \land$$

$$\exists (f \in D_2, dim_{D_2}(f) = 1) : (n, f) \in B_{D_2} \land (n', f) \in B_{D_2}$$

$$G_3(N_3, E_3) \overset{def}{=} \{ n \in N_3 \iff (n \in D_3) \land (dim_{D_3}(n) = 3) \}$$

$$(n, n') \in E_3 \iff (n, n' \in N_3) \land$$

$$\exists (f \in D_3, dim_{D_3}(f) = 2) : (n, f) \in B_{D_3} \land (n', f) \in B_{D_3}$$

We call these graphs D-graph. Figure 4.22 illustrates the D-graph of a rectangle. For clarity the Delaunay polygons have been displaced a bit.

![Figure 4.22: The D-graph for a rectangle. Nodes correspond to Delaunay polygons and edges to Delaunay edges](image)

Let us recall some basic definitions from graph theory that can be used in the analysis of the D-graphs. A path is a sequence of distinct nodes which are pairwise connected by edges
4.4 Importance measures in 3D

Definition 10 (path)
The sequence \((n_0, \ldots, n_k)\) is called a path iff
1. \(\forall i, j \in [1, k-1], i \neq j \iff n_i \neq n_j\) and
2. \(\forall i, 0 \leq i \leq k-1: (n_i, n_{i+1}) \in E\)

If a path between \(n_0\) and \(n_k\) exists we write \(n_0 \Leftarrow n_k\).

A cycle is a path with identical first and last nodes

Definition 11 (cycle)
The sequence \((n_0, \ldots, n_k)\) is called a cycle iff
1. \(n_0 \Leftarrow n_k\) and
2. \(n_0 = n_k\)

Definition 12 (connectedness)
A graph is called connected iff \(\forall n_i, n_j \; n_i \Leftarrow n_j\)

Definition 13 (induced subgraph)
The maximal subgraph of \(G\) which contains only a subset \(S \subseteq N\) of the graph's nodes is called the induced subgraph of \(S\) and designated by \(<S>\)
\(<S> = (S, E'), E' = E \cap (S \times S)\)

Let us introduce also two basic operations on the D-graph which will be used later. The first one is the neighborhood function \(\text{adj}(n)\). It returns the set of nodes which share an edge with node \(n\)

\[\text{adj}(n) = \{n_i | (n, n_i) \in E\}\]

The second one is called \(\text{reachable_subgraph}(n)\) and collects for a given node \(n_0\) and a binary function \(\text{valid}(e)\) all nodes \(n\) which are reachable from \(n_0\) by a path consisting of edges for which the binary function \(\text{valid}(e)\) is affirmative. The pseudocode is given in listing 4.23.

procedure reachable_subgraph(n)
1. reachable_nodes = make_set()
2. processed_nodes = make_set()
3. propagation_front = make_stack()
4. add(n, reachable_nodes)
5. push(n, propagation_front)
6. WHILE size(propagation.front) > 0 DO
7. \(n' = \text{pop}(\text{propagation_front})\)
8. FOR each node \(n, \text{adj}(n')\) DO
9. IF NOT member\((n, \text{processed_nodes})\)
10. add\((n', \text{processed_nodes})\)
11. IF valid\((n', n_i)\)
12. add\((n_i, \text{reachable_nodes})\)
13. push\((n_i, \text{propagation_front})\)
14. RETURN reachable_nodes

Figure 4.23: Pseudocode of the function reachable_subgraph(n).
The subpart mapping

On the elements of the D-graph we can now define the subpart mapping $\mu_2$ in 2D and $\mu_3$ in 3D. We will denote it simply by $\mu$ when both the 2D and the 3D case are meant. The subpart mapping assigns to every edge of the D-graph a set of nodes. Formally it maps the set of edges to the powerset of nodes

$$\mu_2 : E \mapsto 2^N$$

The set of nodes $\mu(e_i)$ assigned to an edge $e_i = (n, n')$ stands for a set of Delaunay polygons in 2D and a set of Delaunay polyhedra in 3D. Hence $\mu(e_i)$ defines a subpart of the object. On this subpart any importance measure $\Delta(\mu(e_i))$ can be defined. In 2D one importance measure has been e.g. the length of the subpart’s outline which is on the object’s boundary. For the following considerations it is only necessary that such a measure is defined but not which specific one it is.

To characterize the subpart mapping let us formulate three basic conditions that the set of nodes $\mu(e_i)$ must fulfill

\[ (C_0) \text{ It must contain one of the two nodes of } e_i. \text{ This means that the subpart of the object assigned to a Delaunay edge/face must be adjacent to that edge/face.} \]

\[ (n \in \mu(e_i) \land n' \notin \mu(e_i)) \lor (n \notin \mu(e_i) \land n' \in \mu(e_i)) \]

\[ (C_1) \text{ The importance measure must be smaller on the set of nodes } \mu(e_i) \text{ than on the remaining nodes. This means that the subpart should not be more important than the rest of the object. Otherwise one should just take the rest of the object as subpart.} \]

\[ \Delta(\mu(e_i)) \leq \Delta(N \setminus \mu(e_i)) \]

\[ (C_2) \text{ The subgraphs induced by the set } \mu(e_i) \text{ and its complement } N \setminus \mu(e_i) \text{ must both be connected. This means that the subpart mapping divides the object into exactly two components.} \]

Under the assumption that the importance measure $\Delta$ is never the same for the subpart and its complement we can prove a theorem which in 2D plays a key role for the unique definition of the subpart mapping $\mu_2$. The assumption is not restrictive since in the case where the two parts of the object are equally important it makes no difference which one of the two parts is chosen as subpart.

**Theorem 2** If $G$ is an acyclic graph the conditions $(C_0)$, $(C_1)$ and $(C_2)$ define a unique subpart mapping $\mu$.

**Proof** by contradiction. Assume that two different subpart mappings $\mu, \mu'$ exist. This means that for at least one edge $e_i = (n_0, n_1)$ the mappings must be different, i.e. $\mu(e_i) \neq \mu'(e_i)$. Then there must exist a node $n \in \mu(e_i)$ for which holds $n \notin \mu'(e_i) \cap \mu(e_i)$. We distinguish two cases:

*case I $\mu'(e_i) \cap \mu(e_i) \neq \emptyset$

The fact that in a connected acyclic graph two nodes are connected by a unique path can be used. Due to condition $(C_0)$ one of the two nodes of $e_i$ must be contained in $\mu'(e_i)$. Assume without loss of generality that this is the node $n_0$. Then $n_1$ is in the complement of $\mu'(e_i)$ due to the condition $(C_0)$. Since the graph has no cycles there exists a unique path $p_{n_0n_1}$ from $n$ to $n_1$ and a unique path $p_{n_0n_0}$ from $n$ to $n_0$. It can be easily seen that $p_{n_0n_1}$ must contain $n_0$. Otherwise the cycle $(n_1, \ldots, n, \ldots, n_0, n_1)$ would exist since the edge $e_i = (n_0, n_1)$ is a trivial path from $n_0$ to $n_1$. Hence the only path from $n$ to $n_1$ contains $n_0$. Since we know that both $n$ and $n_1$ belong to the complement of $\mu'(e_i)$ but $n_0$ belongs to $\mu'(e_i)$, we conclude that the complement of $\mu'(e_i)$ is disconnected which contradicts condition $(C_2)$. 

...
4.4 Importance measures in 3D

Case II $\mu'(e_i) \cap \mu(e_i) = \emptyset$ We show that in this case every node $n$ must belong either to $\mu(e_i)$ or to $\mu'(e_i)$, i.e. the nodes of the graph are partitioned into two distinct sets. Again we denote the node of $e_i$ which is contained in $\mu(e_i)$ by $n_0$ and the other by $n_1$. From condition (C0) and the fact that $\mu'(e_i)$ and $\mu(e_i)$ are distinct we conclude that $n_1 \in \mu'(e_i)$. Every node $n$ is connected to $n_0$ by a unique path $p$ which can have one of the following structure

1. $p$ contains $n_1$, i.e. $p = (n, \ldots, n_1, n_0)$. Then the node $n$ must belong to $\mu'(e_i)$.
   Otherwise $\mu(e_i)$ would contain the two points $n$ and $n_0$ which are only connected by the path $p$ which contains also a node of $\mu'(e_i)$ and the set $\mu(e_i)$ would be disconnected.

2. $p$ does not contain $n_1$, i.e. $p = (n, \ldots, n_0)$. Since we know that the only path between $n_0$ and $n_1$ is $(n_0, n_1)$ we can conclude that the only path between $n$ and $n_1$ is $p \cup (n_0, n_1)$. Then for the same reason as above $n$ must belong to $\mu(e_i)$.

Hence we have shown that $\mu(e_i)$ and $\mu'(e_i)$ partition the D-graph. This means that

$$N \setminus \mu(e_i) = \mu'(e_i) \quad \text{and} \quad N \setminus \mu'(e_i) = \mu(e_i)$$

From condition (C1) we can derive the following two inequalities

1. $\Delta(\mu(e_i)) \leq \Delta(N \setminus \mu(e_i))$ and therefore $\Delta(\mu(e_i)) \leq \Delta(\mu'(e_i))$

2. $\Delta(\mu'(e_i)) \leq \Delta(N \setminus \mu'(e_i))$ and therefore $\Delta(\mu'(e_i)) \leq \Delta(\mu(e_i))$

Hence $\Delta(\mu(e_i)) = \Delta(\mu'(e_i))$ and if we substitute again $\mu'(e_i)$ with $N \setminus \mu(e_i)$ we get $\Delta(\mu(e_i)) = \Delta(N \setminus \mu(e_i))$ which contradicts the assumption that the importance measure $\Delta$ is never the same for the subpart and its complement.

Definition of the subpart mapping in 2D

In 2D two different types of D-graphs can be distinguished according to the topological structure of the object they represent

1. The object is homotopically equivalent to a point. Then it has one single boundary and the D-graph is acyclic.

2. The object is not homotopically equivalent to a point. Then the object has multiple boundaries and the D-graph contains cycles corresponding to Delaunay elements dual to the topological skeleton of the object. All parts of the D-graph which do not represent elements of the topological skeleton are acyclic.

This means that unless it belongs to a cycle of the D-graph every edge in the graph represents a Delaunay edge which cuts the Delaunay triangulation exactly into two parts. The less important of the two object parts is then the subpart associated by the subart mapping to this edge. Hence for such edges the subpart mapping is unambiguously defined by the conditions (C0) - (C2).

The edges which belong to a cycle of the D-graph can be treated in a simple way since they represent the topological skeleton of the object which anyway cannot be deleted when homotopic equivalence is required. So the subpart mapping can remain undefined for these edges and their importance is set to infinity.

Definition of a subpart mapping in 3D

In 3D the situation is completely different. The D-graph has a net-like structure, i.e. it is full of cycles which have nothing to do with the topological structure of the object. The left image of figure 4.24 shows that already for the D-graph of a small box. The subpartner mapping we have to define partitions the nodes of the graph for every edge $e_i$ into two sets. Where these two sets meet there will be edges that have one node in the first set and the other in the second one. Obviously $e_i$ is such an edge due to condition (C0). Note that in an acyclic graph $e_i$ is the only
edge of this type. In 3D where the D-graph contains a lot of cycles there will be many such edges for each possible partitioning of the graph. Indeed there exists a theorem in graph theory that has been formulated in 1927 by Menger [116]. It says that the maximum number of paths joining two points of a graph equals the minimum number of lines needed to separate them. The number of possible paths between two points increases with the number of cycles in the graph. We call the set of these edges for one such partitioning a cutting of the graph. The fact that a cutting in general involves several edges nicely reflects the fact that usually a single Delaunay face does not cut the Delaunay triangulation into two separate parts. One possible cutting of the D-graph of the box is shown in the right image of figure 4.24.

Summarizing we can state that the connectivity of the D-graph in 3D is such that the conditions (C0)-(C2) formulated for the definition of a subpart mapping are not sufficient to define a unique subpart mapping in 3D. Hence the problem is underconstrained. Note that in 2D the condition (C2) guaranteed that the topological structure of the object corresponding to the D-graph was the same with and without the subpart. In 3D this is no longer true due to the more complex topological structure of objects in 3D. Just think of an object with handles where the deletion of a simply-connected subpart is sufficient to remove the handle. Hence condition (C2) must obviously be adapted to guarantee homotopical equivalence between the object with and without the subpart. Another question is whether homotopical equivalence implies also that the boundary of the subparts must be composed of exactly two simply-connected components, one lying on the boundary of the object and one cutting through it. If not, this would be another additional constraint. Note that this question has already been posed in section 4.2 where we asked whether the deletability criterion is mandatory or only sufficient to guarantee homotopical equivalence during regularization. However the problem is still underconstrained and additional constraints have to be found to get a unique subpart mapping.

We have tried several ways to add such additional constraints, but up to now these are based on heuristics and no natural way has been found to define them. It follows a brief description of these attempts.

**Figure 4.24:** *Left figure: The Delaunay triangulation of a small box and its corresponding D-graph. Right figure: A subset of Delaunay faces cutting the D-graph into two parts.*

**Exhaustive search** A brute force approach would be to generate for each edge of the D-graph all possible subparts and then choose the 'best' one. Apart from the fact that this would be computationally very expensive, it just defers the problem to the question what criterion
4.4 Importance measures in 3D

should be used to automatically select the 'best' subpart. The most intuitive answer would be to select the one which has minimal importance. Thereby for the importance of a subpart one of the possibilities described in the next section could be used. However this could lead to counterintuitive results as a simple example may show. Think of an object similar to a dumbbell (cf. figure 4.25). The object can be decomposed into two large parts approximating two balls and a small part connecting the two balls. The D-graph of this connection could roughly be approximated by a net-like planar surface. For any of the edges of this part of the object’s D-graph a subpart can be defined which contains only Delaunay polyhedra of the connection itself and thus has a small importance value. However at least one of the edges must have a big importance value expressing the fact that one of the two balls depends on it.

Figure 4.25: For any edge of the D-graph of the middle part of this object, a subpart (sketched by the hatched region) can be defined which is the 'best' one with respect to the global measures generalized from 2D to 3D but which does not express the importance of the edge for the object's overall shape.

Orientation of the D-graph Another possibility is to assign an orientation to every edge of the D-graph so that it becomes a directed graph. Then for each node \( n_i \) a set of so called preceding nodes can be collected. A node \( n' \) is called a preceding node of \( n_i \) (written \( n' \prec n_i \)) if a directed path from \( n' \) to \( n_i \) exists. To determine the preceding nodes for a given node \( n_i \) the function `reachable_subgraph()` which we have defined for the D-graph can be used with the following condition function:

\[
valid(n_0, n_1) = \begin{cases} 
1 & \text{if } n_1 \prec n_0 \\
0 & \text{otherwise}
\end{cases}
\]

The key question in this approach is how to define an orientation of the graph that complies with the following requirements:

- it should be reproducible and produce similar results for similar D-graphs.
- it should assign to every node a set of preceding nodes on which this node somehow depends.

A possible orientation of the edges is given by the processing orders that have been defined in section 4.3. They assigned an order number to each Delaunay polyhedron which are represented by the nodes in the D-graph. Thus the edges can be oriented so that they point from the node with lower order number to the one with higher order number. As we have seen in section 4.3 different processing orders can be defined. When using 'topological peeling' or a similar processing order one additionally has to define which orientation should be given to an edge joining two nodes of the same order. In principle there are three possibilities

1. No orientation is given to the edge, i.e. the edge is treated as if it wasn't there.
2. Both orientations are given to the edge, i.e. the edge is treated as two edges with opposite orientation.
3. An additional rule is given which selects one of the two possible orientations.
The first solution is too restrictive. It happens that deep in the object there are nodes which have only nodes of the same or higher order in their neighborhood. As a consequence these nodes would have no preceding nodes and the subparts would contain only the Delaunay polyhedron represented by the node itself.

The second solution is obviously too loose since whole layers of polyedra with lower order number will be attached to each polyhedron of higher order. The situation is illustrated in the left image of figure 4.26.

In the third solution various rules can be chosen. Perhaps the most simple one is to define the orientation just randomly. In this case, however, the reproducibility is not given. Otherwise the processing order itself could be refined with some additional condition. The processing orders defined in section 4.3 could be refined by ordering the polyhedra within the same layer according to their distance to the center of gravity of \( D_t \). The center of gravity \( C_{\text{grav}} \) is computed on the Delaunay vertices of the object

\[
C_{\text{grav}} = \frac{1}{n} \sum_{i=0}^{n-1} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}
\]

and the distance \( \delta(p_k) \) of the polyhedron \( p_k \) to \( C_{\text{grav}} \) is defined as the distance between the center of gravity of \( p_k \) to \( C_{\text{grav}} \)

\[
\delta(p_k) = \left\| \frac{1}{|p_k|} \sum_{v_i \in p_k} \begin{pmatrix} v_{ix} \\ v_{iy} \\ v_{iz} \end{pmatrix} - C_{\text{grav}} \right\|
\]

Since the Delaunay triangulation is stepwise reduced the center of gravity of the remaining triangulation can change. Each time a polyhedron \( p_k \) is deleted, there are eventually some vertices \( \{v_0...v_k\} \) which are deleted together with \( p_k \). The new center of gravity \( C'_{\text{grav}} \) can then be computed from the old one:

\[
C'_{\text{grav}} = \frac{1}{n-k} \left( nC_{\text{grav}} - \sum_{i=0}^{k} v_i \right)
\]

The refined processing order can still contain some polyhedra with the same order number. However most edges with nodes of the same order have been eliminated and the remaining ones can be treated as bidirectional edges.

However all three solutions have the common problem that they do not guarantee that the object with and without the defined subparts is always homotopically equivalent. For a single subpart this could be achieved by iteratively checking whether all polyhedra represented by the subpart can be deleted. If they are not one could also try to adapt the generated subpart by selectively flipping the orientation of edges in the D-graph. While this is already difficult for a single subpart it is probably impossible to achieve for all subparts together since the flipping of an edge in the D-graph leads automatically to a change in more than one subpart. On the right two images in figure 4.26 a front and a back view of such a subpart which has been defined using an oriented D-graph and which cannot be removed are shown. From our experience we can say that such cases are rarer when a refined processing order is used, however they cannot be excluded.

**Cone of View** A third approach is more related to the local measure of the angle between the meeting firefronts. Let us explain it on behalf of figure 4.27. For each node \( n_{\text{max}} \) in the D-graph we select the edge \( e_i = (n_{\text{max}}, n_{\text{min}}) \) for which the second node \( n_{\text{min}} \) has minimal processing order number. By this choice it is most likely that the edge is oriented towards the object's boundary. The edge \( e_i \) represents a Delaunay face \( f_i \) which corresponds to a Voronoi edge. The straight line defined by this Voronoi edge intersects \( f_i \) at a point \( P \). The Delaunay face is composed of a number of vertices. We arbitrarily select one of them, e.g. \( v_0 \). Then the points \( P, n_{\text{max}} \) and \( v_0 \) form a triangle \( \Delta_{P,n_{\text{max}},v_0} \). By rotating this triangle around the edge \( e_i \) a cone is generated. We
call it the cone of view of the Voronoi edge since it defines the field of view for an imaginary observer which sits on the Voronoi edge at $n_{\text{max}}$ and looks towards $n_{\text{min}}$ so that he can just barely see the vertices of $f_i$.

Figure 4.27 is a sketch of the so defined cone of view. Its aperture is defined by the angle

$$\alpha = \arccos\left(\frac{s \cdot h}{|s| \cdot |h|}\right)$$

Hence all points $P_i$ inside the cone of view must fulfill the following property:

$$\frac{s' \cdot h}{|s'| \cdot |h|} \geq \frac{s \cdot h}{|s| \cdot |h|} \tag{4.1}$$

The cone of view is then used to generate a subpart for $n_{\text{max}}$ in the following way:

1. a set $S_0$ of all nodes lying inside the cone is generated.

2. a set of nodes $S_1$ is iteratively defined starting from $n_{\text{max}}$ and including all the nodes connected to $n_{\text{max}}$ by a path containing only nodes from the set $S_0$. 

Figure 4.26: \textit{Left figure:} Subpart assigned to the edge $e_i$ when edges between two nodes of the same order are taken to be bi-directional. \textit{Right two figures:} Front and back view of a subpart which cannot be removed from the object if homotopical equivalence between the object with and without the subpart is required.
The set $S_0$ is computed by checking for all nodes whether they fulfill equation 4.1 or not. To determine $S_1$ the function `reachable_subgraph()` is used with the condition function `valid_0()`.

$$\text{valid}_0(n_0, n_1) = \begin{cases} 1 & \text{if } n_0, n_1 \in S_0 \\ 0 & \text{otherwise} \end{cases}$$

The set of nodes $S_1$ represents then the subpart assigned to $e_i$. Note that the set $S_1$ may be smaller than the set $S_0$ since there can be several disconnected parts of the object lying inside the cone of view.

**Figure 4.27**: The cone of view under which an imaginary observer sees the boundary face $f_i$ when sitting on the D-graph at node $n_{max}$.

**Figure 4.28**: The boundary faces intersected by the cone of view (indicated by the red lines) of a node on the part of the D-graph corresponding to the conical skeletal face which represents the upper part of the cylinder in the original object.
In figure 4.28 the cone of view assigned to a Voronoi vertex of the skeleton of the cylinder merged with a box is shown. We have drawn only the Delaunay faces on the boundary of the object which are intersected by the cone of view (sketched by a bundle of straight lines). In this case the cone of view leads to an appealing result. However there are many cases, especially for small polyhedra near the boundary, where the cone of view becomes much too wide and consequently assigns large subparts to insignificant polyhedra.

Once a subpart mapping has been established, different global importance measures can be defined to express the significance of these subparts with respect to the rest of the object

• The ratio between the area of the subpart’s boundary lying on the object’s boundary and the area of the subpart’s boundary cutting through the object could be used. Since the part of the subpart’s boundary which cuts through the object can be fairly complex (it is not a single Delaunay element as in 2D!) the question must be formulated whether the removing of the subpart from the object can be regarded as a simplification of the object.

• The area of the subpart’s boundary lying on the object’s boundary could be compared with the overall area of the object’s boundary. Such a measure would give a high importance to small protrusions on the object’s boundary. Such protrusions have a large boundary area enclosing only a small volume. One can take the view that such protrusions in a shape are highly significant as well as one can hold that they do not really contribute to the overall shape. A 2D example for such an object is shown in figure 4.29.

• The same comparison could be made between the subpart’s volume and the object’s volume.

Or, what seems to me preferable, to compare the two volumes defined by the subpart and by the rest of the object that remains when the subpart has been removed. In order to determine these subvolumes the regeneration procedure described in chapter 5 could be used.

One should be aware that the above described attempts to define global importance measures basically rely on heuristics which is a bit unsatisfactory. Moreover their generation is very costly limiting their applicability in practice. To justify the computational overhead involved in the determination of global importance measures they must fulfill two requirements

• They should be able to assign a subpart of the object to skeletal parts which is in accordance with our visual perception. This is the key to automatically establish a shape hierarchy.

• In 2D some of the global measures have been proved to be monotonic. As a consequence the regularization became a simple thresholding. Since in 3D no partial ordering exists on the D-graph one cannot speak about monotonicity as in 2D but nevertheless a global
measure in 3D should also guarantee the preservation of the topological structure if simple thresholding is applied.

However at the present moment there is no such measure in sight and local measures are used in practice. We will see in the next chapter on the postprocessing that after regularization the remaining Voronoi faces can be grouped into larger unbranched skeletal sheets based alone on neighborhood relations between the Voronoi faces. In general a small number of such sheets will be sufficient to describe the skeleton of an object. To my opinion it would be much more promising to search a global importance measure for these skeletal sheets and not for single Voronoi faces. In other words one should use local measures to get rid of most of the insignificant parts of the Voronoi diagram and forget about using global measures at this early stage of regularization. Global measures could eventually be reconsidered on the regularized Voronoi diagram where they would still be the key to automatically establish a shape hierarchy.

4.5 The dimensionality of skeletal constituents

The definition of the SAT by the centers of all maximal balls inscribable to the object showed the close relationship between the distance map of the object and its skeleton. Thereby we have said in section 2.1 that the skeleton corresponds to ridges in the distance map. The dimensionality of these distance ridges corresponds to the different types of local symmetry represented in the 3D skeleton. These are (cf. section 1.3)

- In the degenerate case when the object is a sphere a single Voronoi vertex will represent the object's rotational symmetry.
- Skeletal lines composed of Voronoi edges represent axial symmetry.
- Skeletal sheets composed of Voronoi faces represent mirror symmetry.

In most skeletonization techniques, e.g. thinning, the type of symmetry to detect is chosen a priori. This choice can only be made appropriately if the object to be skeletonized is known in advance, i.e. whether the object is better described by axial or mirror symmetries. In medicine for example one would prefer the representation of axial symmetries for vessel trees and the representation of mirror symmetries for most organs. However it is questionable whether only one type of symmetry is sufficient to represent general 3D objects. Therefore the type of symmetry should be decided locally by the skeletonization procedure itself.

Such a local decision can be well done by using the connection of the skeleton with the object's distance map. In fact the type of symmetry can be well detected in the object's distance map by the differential geometric characterization of the distance ridges as presented in section 2.1. The analysis is done with the aid of the Hessian matrix defined in section 2.1 and which can be estimated locally from the distance map. The eigenvectors of the Hessian matrix form a local coordinate system which is oriented in the principle axis of the ridge. The eigenvalues $\lambda_1, \lambda_2, \lambda_3 \leq 0$ of the negative Hessian matrix can be used to discriminate between the different symmetry types

- rotational symmetry is given where $\lambda_1 \approx \lambda_2 \approx 0$,
- axial symmetry where $\lambda_1 \approx 0 > \lambda_2$ and
- mirror symmetry where $\lambda_1 \approx 0 > \lambda_2$.

This gives us a measure to locally control the dimensionality of skeletal constituents according to the symmetry type represented in the original object. In figure 4.30 the Voronoi edges of the unregularized Voronoi diagram of a cylinder are shown. The colour indicates the strength of axial symmetry found in the discrete distance map.
Figure 4.30: The Voronoi edges of a cylinder coloured according to their symmetry type. The red colour indicates Voronoi edges in the vicinity of voxels of the distance map which represent axial symmetry.
Chapter 5

Postprocessing

5.1 From Voronoi faces to skeletal sheets

The careful approximation of the original object by sampling its boundary at voxel level introduced a huge number of Voronoi elements which did not belong to the object's skeleton and hence have been eliminated during the regularization. As a consequence the Voronoi skeleton consists of many small Voronoi faces which in the following we will call elementary faces. The high number of elementary faces is prohibitive for most applications. Hence a postprocessing is needed to aggregate them into groups representing unbranched skeletal parts. We call this postprocessing step grouping. The face groups are produced sewing elementary faces together at edges where only two faces meet.

The grouping algorithm first selects as a startpoint a face \( f_0 \) which does not yet belong to a group. Then it checks all adjacent faces \( f_i \) which have not yet been grouped. Each of them shares an edge \( e_i \) with the face \( f_0 \). If this edge bounds no other faces than \( f_0 \) and \( f_i \), the face \( f_i \) can be added to the group. This is repeated for each face which has been added to the face group until no more faces can be added. Then the next ungrouped face is selected and the grouping starts over again. The algorithm stops when all faces have been assigned to a group. The procedure `face_grouping()` described by the pseudocode in listing 5.1 implements this algorithm. The function `get.next(<list>)` extracts the next element from a list. Let us explain the procedure with the help of figure 5.2:

lines 1-2 All faces are put into the list of ungrouped faces.

lines 4-5 The aggregation of the next group begins here. An arbitrary face is selected from the list of ungrouped faces and put into a new list of not yet processed faces which belong to the new group. In figure 5.2 the faces currently in this list are marked with the same pattern.

lines 7-8 From the list `faces.to.process` a face \( f_i \) is selected. It receives the id of the new group. All faces which have already been assigned their group id are again marked with the same pattern in figure 5.2.

lines 9-17 The edges of \( f_i \) which bound exactly one other face besides \( f_i \) (line 13) that does not yet belong to a face group (line 16 and 17) are selected. In figure 5.2 these edges are drawn as thick dashed lines if they bound only two faces and as thick solid lines if the second face they bound is not yet in list of faces to process. The new faces which are bounded by these edges are added to the list of faces to process.

line 18 The next group with a new id is determined.

This grouping algorithm is straightforward. However some undesired situations can arise as shown in figure 5.3. In the shown case all faces are aggregated into one single group. The arrows show a possible sequence in which the faces could have been processed by the grouping algorithm. A human operator probably would have put the faces into two different groups. Face groups of
procedure face_grouping()

1. ungrouped_faces = make_list()
2. FOR ALL faces \( f_i \) DO
   insert\( (f_i, \text{ungrouped_faces}) \)
END FOR
3. WHILE size(ungrouped_faces) > 0 DO
4. \( f_0 \) = get_next(ungrouped_faces)
5. faces_to_process = make_list\( (f_0) \)
6. WHILE size(faces_to_process) > 0 DO
7. \( f_i = \text{get_next}(\text{faces_to_process}) \)
8. group\( (f_i) = \text{group_id} \)
9. edges_to_process = incident_edges\( (f_i) \)
10. WHILE size(edges_to_process) > 0 DO
11. \( e_i = \text{get_next}(\text{edges_to_process}) \)
12. neighbor_faces = incident_faces\( (e_i) \)
13. IF size(neighbor_faces) == 2
14. \( n f_1 = \text{get_next}(\text{neighbor_faces}) \)
15. \( n f_2 = \text{get_next}(\text{neighbor_faces}) \)
16. IF \( (f_i == n f_1) \)
   IF member\( (n f_2, \text{ungrouped_faces}) \)
   insert\( (n f_2, \text{faces_to_process}) \)
   ELSE
   IF member\( (n f_1, \text{ungrouped_faces}) \)
   insert\( (n f_1, \text{faces_to_process}) \)
END WHILE
END WHILE
18. group_id = group_id + 1
END WHILE

Figure 5.1: Pseudocode of the face grouping

this type can be detected since they contain edges which bound more than two faces of the same group. In the example of figure 5.3 these edges are shown as thick lines. Hence the problem can be resolved by splitting face groups of this type in the following way:

1. Edges bounding more than two faces of the same group \( g_i \) are detected.
2. New group numbers are assigned to the faces which join along these edges.
3. These group numbers are then propagated to the neighboring faces until all faces of group \( g_i \) have received a new group number, i.e. group \( g_i \) is split into a number of new groups.

The result of the grouping process on the skeleton of a box merged with a cylinder is shown in Figure 5.4. Each face group has been assigned a different colour. As expected the numerous Voronoi faces are now aggregated into a few macroscopic skeletal sheets.

Especially at the boundary of the shape the Voronoi diagram tends to produce many small faces as we have seen during the regularization. Moreover the local significance measures have difficulties to clearly discriminate between skeletal faces and faces produced by discretization effects apart from the fact that even the exact skeleton of complex shapes can become very detailed at the object boundary. These small faces are often multiply branched and therefore cannot be grouped into larger groups of faces by the above described process. The consequence is that even after the grouping procedure has been applied to the approximate skeleton of such
5.1 From Voronoi faces to skeletal sheets

H grouped facets
in list facets_to_process

Figure 5.2: Illustration of the successive formation of unbranched skeletal sheets by the grouping of elementary Voronoi faces.

Figure 5.3: A skeletal sheet produced by the grouping procedure which contains a branching line (bold line). The arrows indicate a possible sequence of face aggregation, for which the branching line cannot act as a barrier stopping the grouping procedure.

objects, many insignificantly small face groups will persist. In order to get rid of them yet another pruning stage can be introduced. This time the pruning is performed on entire face groups and the significance measure will be either the area of the face group or the volume of that part of the object which can be reconstructed from the face group. After such a pruning stage the face aggregation procedure can be repeated since there will be now additional Voronoi edges joining only two Voronoi faces.
5.2 Regeneration of the shape

Obviously once we have the skeleton of an object and use it as shape descriptor we want to be able to regenerate the original object from its descriptor. Hence we are interested to know what is the regeneration error that we must expect. First of all let us describe how the original object can be regenerated from its skeleton.

From the definition of the skeleton we know that it represents maximal balls inscribed to the object. Hence to regenerate the original object it suffices to draw all the maximal balls represented in the skeleton. The radius $r$ of a maximal ball $B_{\text{max}}(x, r)$ to be drawn is given by the distance of the skeletal point $x$ to one of the boundary points at which a fire-front that quenches at $x$ has been originated. In the Voronoi skeleton each skeletal part corresponds to a Voronoi element (vertex, edge or face) and has a reference to its dual counterpart in the Delaunay triangulation. The Delaunay elements are made up of Delaunay vertices which represent boundary points of the object. Hence for each skeletal part we dispose of at least one boundary point. The calculation of the maximal ball's radius is then straightforward.

As we have seen the proximity information encoded in the Voronoi diagram gives us the possibility to decide for every point on the Voronoi skeleton which boundary point (out of the discrete set of points by which the object has been represented) is the nearest one and consequently to determine the maximal ball for every point on the Voronoi skeleton. However we don't really need all maximal balls (which would be infinitely many) to regenerate the object. It is sufficient to use the maximal balls belonging to the Voronoi vertices because the maximal balls of other Voronoi elements are contained in the union of those maximal balls which belong to the Voronoi vertices of that Voronoi element. A proof that this property is true can be sketched in the following way:

1. First show that the property is true for a Voronoi edge. This is not difficult as the left
5.3 The hierarchical skeleton

Figure 5.5: Illustration of the proof that the regeneration procedure needs only to take into account the maximal balls of the Voronoi vertices and not of intermediate points on Voronoi edges or faces.

part of figure 5.5 illustrates. The three maximal balls \( B_0(v_0, r_0), B_i(v_i, r_i), B_1(v_1, r_1) \) all go through the same boundary point \( p \) and in the figure we show the plane defined by the Voronoi edge and the boundary point. Obviously the radius \( r_i \) is between \( r_0 \) and \( r_1 \). The projection of the maximal balls into the plane are great circles of the balls and the property can be directly seen from the figure.

2. If the property holds on Voronoi edges, it holds also on the Voronoi faces bounded by these edges. The proof is shown on the right part of figure 5.5. We simply choose a Voronoi vertex \( v_0 \) of the face’s boundary and consider the line passing through \( v_0 \) and the face point \( v_j \). Since the Voronoi face is convex this line will intersect one of the bounding Voronoi edges, say \( v_kv_{k+1} \) at a point \( v_i \). Since the property holds for Voronoi edges we know that

\[
B_i \subset \bigcup \{B_k, B_{k+1}\}
\]

and that

\[
B_j \subset \bigcup \{B_0, B_i\}
\]

Then we conclude that

\[
B_j \subset \bigcup \{B_0, B_k, B_{k+1}\}
\]

The regeneration of the original object from the discrete set of maximal balls of all Voronoi vertices in the inner of the object is shown in figure 5.6 for the artificial object of a box merged with a cylinder. On the left the original object as it comes from the 3D image raster can be seen, in the middle the regeneration with continuous balls is shown. The irregular surface is an effect of discretization. In fact the original object has been defined by a discrete set of points. Hence the regeneration of the object can only be expected to exactly reproduce the sampling points of the original object, the behaviour of the boundary in between them will depend on the regeneration method used and can vary within half the sampling density. This discretization effect has no visible consequence in practice since when the maximal balls are also discretized by the same image raster as the original object the regenerated object will be exactly the same as the original one. On the right of figure 5.6 one can see the regeneration of the original object from the Voronoi skeleton which resulted after the pruning of the Voronoi diagram. There are only a few voxels of the original object missing.
5.3 The hierarchical skeleton

Ogniewicz [128, 130] showed that the 2D Voronoi skeleton can be organized in a hierarchical manner. Monotonic significance measures can be used to establish different hierarchy levels of the skeletal branches. In figure 5.7 we show the value of such a measure for an artificial object as the height of sheets positioned at the skeletal branches. The significance measure defines a Voronoi vertex with the highest measure value, clearly visible in the figure. It is reasonable to select this vertex as the topological skeleton of the object. Starting from this point, the Voronoi edges can be traced consecutively along the skeleton in both directions. At every branching points the path with the larger residual value can be followed. This way, a single, non-branched skeletal line can be extracted, shown as a bold line on Figure 5.8. This skeleton part, which is called the first order skeleton represents the most dominant aspects of the object shape. In degenerate cases where nearly or completely equivalent skeletal branches join, the first-order
skeleton may be branched, too.

In the following step the branches neglected in this first path are processed and followed similarly in resulting a set of second order skeletal branches. Recursive processing generates a complete hierarchy between skeletal branches illustrated on figure 5.8. The depth in the hierarchy is coded as the width of the skeletal lines. The first order branching points (between the first order skeleton and its second order branches) are denoted as filled circles.

The skeletal hierarchy is a very powerful tool for addressing the two different aspects of shape depending on the particular application as analyzed in the introduction. Lower order skeletal branches represent global shape features and can be selectively used for object matching. On the other side, the higher order branches describe the fine, local variations of the individual objects and can be efficiently used for their unique characterization. This way the hierarchical skeleton can fulfill the contradictory requirements of the different tasks as a single shape descriptor by unifying coarse and fine features in a single, organized data structure.

The generation of the skeletal hierarchy relies on two important properties of the 2D skeleton:

1. it's topological structure, i.e. that the branches attached to the topological skeleton are hierarchically ordered.

2. the definition of monotonic significance measures.

In 3D the automatic generation of a skeletal hierarchy is obstructed by the missing of both these properties as we have seen in the chapter 4 on regularization of the Voronoi diagram. Up to now no practically applicable procedure for the automatic generation of a unique hierarchy of the skeletal constituents in 3D has been found. It may even be questioned if the definition of such an algorithm is theoretically possible at all. Nevertheless in many cases a skeletal hierarchy can be established manually. For that purpose we have developed interactive tools which allow us to navigate in the graph structure of skeletal sheets, to further aggregate the groups of Voronoi faces as produced by the grouping procedure into large unbranched skeletal sheets and finally to define a hierarchy among these skeletal sheets. This hierarchy can give us later a mean to distinguish between coarse and fine features of the shape and is used for example to control the level of detail up to which we want to reconstruct a given shape from its skeleton.

The efficiency of this manual working step depends on the quality of the assisting tools. In the following I would like to give a collection of ideas of how the generation of the skeletal hierarchy can be supported by automatic procedures indicating which of them have already been implemented and used for this thesis

- In a first step the face groups are ordered according to their area. Then any face group can be selected by the user. The computer program searches automatically the neighboring groups and presents them to the user. Neighboring groups are defined to be the groups that are attached to at least one of the border edges of the selected face group. The user can then click on one of the neighbor groups and append it to the previously selected group. This way large unbranched skeletal sheets can be iteratively produced.

Figure 5.8: Skeleton of the artificial object after pruning
5.3 The hierarchical skeleton

During this further aggregation we realized that the automatic grouping of the Voronoi faces sometimes produces too large groups, especially when it is performed on a set of Voronoi faces as it results after a first grouping with a subsequent regularization on the face groups. The reason for this is that the insignificantly small face groups often act as barriers for the grouping process and are responsible for that the face groups don't get too wide spread. Hence a tool for the subdivision of too large sheets has been developed.

Up to now the completion of the basic skeletal sheets is done manually by clicking on the right neighboring face groups. This could be further automated by searching for face groups which give a 'smooth' continuation of the group. The smoothness of the transition between two neighboring face groups could be defined by the normals of the Voronoi faces near their common border. To compare the distribution of two sets of normals an extended Gaussian image (EGI) as proposed by [83] could be used. The idea of this representation is to map the normals of the single Voronoi faces onto a point on the unit sphere. One has to be aware of the fact that a Voronoi face can have two opposite normals which are equivalent since skeletons are not closed surfaces but 2D-manifolds for which no inner and outer side is defined. Hence we adapt the EGI by representing each normal with two opposite points on the unit sphere.

Figure 5.9 illustrates the EGI on two skeletal sheets of an example skeleton. One sheet is cone-like and the other approximately represents a plane. Accordingly the EGI of the first sheet resembles a ring of the Gaussian sphere and the second one to a point. One can see that the EGI is both times symmetric due to the representation of both normal directions of each Voronoi face. In the right image the EGI's of three skeletal sheets are merged together to illustrate that the "distance" between the EGI's could be used to find smooth continuations between face groups.

The final step is to assign a hierarchy level to each one of the new face groups. Since this hierarchy should later provide a basis to distinguish coarse shape features from fine ones it makes sense to adopt the following growing strategy from the inside to the outside of the object:

- Select the face group which represents the topological skeleton as the first order skele-
5.4 Illustration of the postprocessing on a synthetic object

We want to illustrate this whole process of establishing a skeletal hierarchy on a synthetic object which has been produced by sweeping a slice through the white brain matter of the temporal lobe in the third dimension. A rendering of the surface of the resulting object is shown in figure 5.10 on the left hand side. The image raster has the size $71 \times 71 \times 71$. The surface has been sampled at voxel level and is represented by 45,294 points. The computation of the Delaunay triangulation and the Voronoi diagram took about 14 minutes on an UltraSparc 167MHz. An overview of the number of Delaunay and Voronoi elements before and after the regularization is given in table 5.1. For the regularization a threshold of 20 degrees has been used. This means that Voronoi faces which represent ridges in the distance map of an angle greater than 130 degrees are possibly pruned away.

The resulting skeleton is shown in figure 5.10 on the right hand side. The grouping of the remaining 30,114 microscopic Voronoi faces produces 2055 face groups. The left graph of figure 5.11 shows the areas of the face groups. With the help of this graph one can see immediately that many of these face groups are insignificantly small and hence will disappear after a second regularization step with the area of the face groups as significance measure. In fact the second regularization which has been applied with a threshold of 1 pixel reduced the number of microscopic Voronoi faces to 28,504 and a subsequent grouping produced 276 macroscopic face groups. In figure 5.12 a different colour has been assigned to the largest face groups while the rest of

Figure 5.10: Left figure: Surface rendering of an artificial object generated by sweeping a slice of the white matter of the temporal lobe in the z-direction. Right figure: Skeleton of the same object coloured according to the ridge-strength importance measure.
Table 5.1: Delaunay and Voronoi elements describing a swept slice of the temporal lobe

The procedures exemplified in this section on a fairly simple object have been also tested on very complex shapes as the human brain and are currently used in different prototypical medical applications. The last chapter is dedicated to theses applications.
Figure 5.12: The face groups resulting from the automatic grouping procedure. The subdivision of the skeleton can be seen from its colouring. The smallest face groups have all been assigned the same colour.
5.4 Illustration of the postprocessing on a synthetic object

Figure 5.13: From left to right (a-d): Manual aggregation of four face groups into one unbranched skeletal sheet which represents the first hierarchy level of the skeleton. The aggregated face groups are shown in blue, their neighboring groups in red. The last two images show all of the 30 manually aggregated skeletal sheets representing the full skeleton. In the left image (e) each skeletal sheet has a different colour. In the right image (f) the sheets are coloured according to which level in the skeletal hierarchy they belong to.

Figure 5.14: Upper row from left to right (a-d): Reconstruction of the original object from the skeleton at four different levels of detail using the manually established shape hierarchy. Lower row from left to right (e-h): The same reconstruction levels but coloured according to the distance from the regenerated shape to the original one.
Chapter 6

Results and applications

The analysis of the enormous theoretical and technical difficulties in the previous chapters and the problems that remain unsolved may create the impression that the generation of 3D Voronoi skeletons is not feasible. However this chapter demonstrates the contrary. In spite of the problems we mentioned practical applications have shown that 3D skeletonization is a powerful tool to describe shape in three-dimensional space.

The first part of this chapter is dedicated to the presentation of 3D Voronoi skeletons of different artificially generated objects. In the second part of the chapter I would like to illustrate the power of skeletal representation on two prototypical medical applications. In the first case bone thickness in the acetabulum has to be investigated for optimal prosthesis placement in hip joint replacement operations. In the second example the generation of skeletons of a human brain extracted from 3D MRI data will be demonstrated. To my opinion these examples reflect the two major benefits that skeletal shape description can offer:

1. it gives us a consistent framework to distinguish coarse shape features from fine ones
2. it has the potential to make explicit certain properties of the shape it represents which otherwise are difficult to perceive.

6.1 Skeletons of artificially created shapes

The skeletons of artificially created objects presented in this section should help to become familiar with skeletons of 3D objects and demonstrate some fundamental properties of the Voronoi skeletons as

- invariance under geometrical transformations as translation, rotation and scaling.
- insensitivity to geometric noise.
- rich local support which means that the skeleton is locally insensitive to changes in the shape which occur far from the current focus of interest

In figure 6.1 a surface rendering of the investigated objects is shown. As test objects we have selected a cylinder, a box and the composition of them. Each of these objects has been generated twice, once aligned to the image raster and once in an oblique position. Table 6.1 presents the original test objects and the result of the Voronoi diagram generation. Figure 6.2 shows the Voronoi diagram for each test object. One realizes immediately that due to discretization effects on the image raster the Voronoi diagram becomes much more complex for the objects in oblique position. In fact the box which has been aligned with the image raster requires almost no regularization. This is also reflected in the statistical summary of the regularization in table 6.2. The faces of the Voronoi diagrams have been coloured according to the ridge-strength measure. In that way it becomes immediately clear that most of the Voronoi faces of the test objects in
6.1 Skeletons of artificially created shapes

Figure 6.1: Surface renderings of the six artificially generated test objects. The objects in the upper row are aligned to the image raster, those in the lower row are skewed by 155° against the xy-plane and 35° against the yz-plane.

<table>
<thead>
<tr>
<th>Test object</th>
<th>image size [voxel]</th>
<th>volume [voxel]</th>
<th># boundary points</th>
</tr>
</thead>
<tbody>
<tr>
<td>box (a)</td>
<td>100x100x100</td>
<td>147'000</td>
<td>18'200</td>
</tr>
<tr>
<td>cylinder (b)</td>
<td>100x100x100</td>
<td>33'480</td>
<td>7'528</td>
</tr>
<tr>
<td>composition (c)</td>
<td>100x100x100</td>
<td>164'280</td>
<td>21'528</td>
</tr>
<tr>
<td>skewed box (d)</td>
<td>100x100x100</td>
<td>142'632</td>
<td>27'800</td>
</tr>
<tr>
<td>skewed cylinder (e)</td>
<td>100x100x100</td>
<td>32'838</td>
<td>9'528</td>
</tr>
<tr>
<td>composition (f)</td>
<td>100x100x100</td>
<td>159'736</td>
<td>31'768</td>
</tr>
</tbody>
</table>

Table 6.1: Data of the six test objects

oblique position (objects d, f) will be pruned away in the regularization step. In table 6.2 the regularization threshold is given for the ridge-strength measure. It can be chosen in the interval [0°, 45°]. A value of 0° means that no regularization takes place one of 45° that the Voronoi diagram is reduced to the topological skeleton of the object.

The resulting skeletons after the regularization are depicted in figure 6.3. The invariance of the skeletons under geometric transformations, especially under rotation is well demonstrated with these images. Only near the object’s boundary, part of the skeleton is lost due to the general weakness of distance ridges near object boundaries. Note that the discretization of the rotated test objects with the image raster introduced geometric noise to the position of the object’s boundary points. Hence the insensitivity of Voronoi skeletons to geometric noise can also be seen on the skeletons of these objects. Finally the rich local support of Voronoi skeletons becomes apparent when one compares the skeletons of the box and of the cylinder with the skeleton of their composition.
6.2 Bone thickness characterization using skeletonization

Optimal placement of hip joint replacement prosthesis requires general knowledge about the thickness of the surrounding bone structures. Such knowledge can be extracted from the analysis of the anatomical variation in a selected training population. Quantitative evaluation, however, is not possible without adequate representation of the hip bone. Skeletal description provides a natural basis for this analysis and proved to be useful for the characterization and visualization of the thickness of the hip bone around the acetabulum.

Figure 6.4 shows a 3D rendering of a hip bone to be analyzed. The dataset contained 49,733 boundary points, which produced 65,073 Delaunay polyhedra and 102,447 elementary Voronoi faces. After a first regularization step using the ridge strength as significance measure 29,928 elementary Voronoi faces were left. The remaining faces have been grouped into 1,218 face

<table>
<thead>
<tr>
<th>Test object</th>
<th>Delaunay triangulation</th>
<th>Voronoi diagram before regularization</th>
<th>Voronoi diagram after regularization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vertices</td>
<td>edges</td>
<td>faces</td>
</tr>
<tr>
<td>box (a)</td>
<td>18'200</td>
<td>45'000</td>
<td>35'426</td>
</tr>
<tr>
<td>cylinder (b)</td>
<td>7'528</td>
<td>22'903</td>
<td>22'144</td>
</tr>
<tr>
<td>composition (c)</td>
<td>21'528</td>
<td>56'503</td>
<td>48'086</td>
</tr>
<tr>
<td>skewed box (d)</td>
<td>27'800</td>
<td>106'936</td>
<td>124'155</td>
</tr>
<tr>
<td>skewed cylinder (e)</td>
<td>9'528</td>
<td>35'990</td>
<td>40'879</td>
</tr>
<tr>
<td>composition (f)</td>
<td>31'768</td>
<td>122'394</td>
<td>141'994</td>
</tr>
</tbody>
</table>

Table 6.2: Number of elements contained in the Delaunay triangulation of the test objects and statistic of the Voronoi diagrams before and after the regularization with t_reg.
groups. An analysis of these groups shows that many of them represent insignificantly small skeletal parts (cf. left graph of figure 6.6). Hence a second regularization step acting on these face groups and using the overall area of the group as significance measure has been done to clean up the skeleton. All groups with an overall area of less than 10 pixels have been removed if homotopical equivalence could be guaranteed. Thereafter 28'196 faces were left. Obviously this produced an additional number of edges joining only two elementary Voronoi faces. Hence the aggregation procedure has been repeated which led to 179 face groups of which only 70 had an area greater than 10 pixels (cf. right graph of figure 6.6). Figure 6.5 shows the resulting skeleton.
6.3 Analysis of the cortical brain structure

The cerebral cortex is the highly convoluted surface of both brain hemispheres. Cortical folds or sulci are deep narrow fissures in the brain that greatly increase the cortical surface. Dual
6.3 Analysis of the cortical brain structure

Figure 6.7: The medial surface representing the acetabulum of the hip bone colored by the local bone thickness (from black to white). The arrows indicate optimal areas for prosthesis support.

to them are the gyri which represent cortical regions delimited by sulci. During the last years several large programs have been initiated in the USA, in Europe and in Japan which aim at the identification of the cortical brain structure and there are currently much research efforts related to the human brain. It is no accident that this decade is sometimes called the “brain decade”. These studies have substantially increased our knowledge about the human brain structure and researchers have begun to manually construct atlases of the human brain structure (ct. for example the work of Höhne et al. [80]).

We would like to identify two complementary goals inside this domain of research where skeletonization could play a major role:

- human brain mapping between different subjects
- analysis and characterization of the cortical structure

The first goal addresses an increasing need in neurology to precisely identify brain structures as fast as possible and hence to develop tools which efficiently assist neuroanatomists in doing so. This need appears especially urgent in the field of human brain mapping, where several programs have been initiated to create an electronic database of human functional neuro-anatomy. It appears also crucial in the design of precise neuro-surgical operations. Indeed, the success of micro-surgical techniques depends on using natural sulcal pathways to gain access to pathological structures within the brain, while preserving the integrity of healthy tissues. Concerning the topography of the human cortex it should be noted that today only experienced neuroanatomists are able to identify the cortical brain structures and that there is no full consensus on what should be the result of such an identification. From the image processing point of view the challenge lies in finding a flexible way to embed inter-individual variability into a common structural model of the cortex topography. Hence one needs to concentrate on the large-scale features of the human cortex which are common between different individuals and to neglect the individual variations in a controllable way. This exactly recalls the properties of the skeletal hierarchy. Although we are far from reaching this goal we strongly believe that skeletonization could be of great utility to develop such a model and provide a good basis in the future for matching hierarchically the model to individual anatomy. We would like to stress that region-based methods as skeletonization seem to be more adapted to describe the shape of the human cortex than surface-based methods since the cortex is more complex and subject to inter-individual variation near the surface of the brain and more uniform deep inside. The skeletal representation of the inner cortical surface we will
compute to address the second goal could be generated in the same way for a carefully segmented brain cortex which then can serve as a model. This will involve a considerable amount of work but it has to be done only once. The actual matching of the patient anatomy to the model could then be made between the model skeleton and the full symmetry set derived directly from the gray-valued MR image of the patient as described in section 1.3. A comparison of the advantages and deficiencies of skeletons on the one hand and the full symmetry set on the other hand shows that these are complementary and thus motivates this approach (cf. table 6.3). This idea is discussed into more detail in [174].

<table>
<thead>
<tr>
<th>Skeleton</th>
<th>Full Symmetry Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ manageable set of symmetry axes</td>
<td>no prior segmentation needed</td>
</tr>
<tr>
<td>hierarchically ordered</td>
<td>robust to topological noise</td>
</tr>
<tr>
<td>- presegmentation needed</td>
<td>vast set of symmetry points</td>
</tr>
<tr>
<td>sensitive to topological noise</td>
<td>lack of connectivity</td>
</tr>
</tbody>
</table>

Table 6.3: Advantages and deficiencies of skeletons on the one hand and the full symmetry set on the other hand.

For the second goal the skeletons of the human cortex allow us to neglect the common and for this goal uninteresting large-scale shape features and to focus on the unique distinctive features. We are currently exploiting this technique in a joint European project called BIOMORPH [43] whose scope is to develop improved techniques for measurement of the size and shape (morphometry) of the brain and its parts and to apply and validate them in the context of certain diseases as schizophrenia. Neuroanatomical and histological findings from post-mortem brains, as well as in vivo findings from MRI studies suggest the presence of morphological temporal lobe abnormalities in schizophrenia. To determine, whether or not sulco-gyral pattern abnormalities in the temporal lobe could be detected in vivo, computerized surface rendering techniques for MR data have been developed in order to make qualitative and quantitative analysis of three-dimensional reconstruction of the temporal and frontal cortex. 3D renderings in the same standard orientation of the brain surface have been used to determine characteristics of the sulco-gyral patterns correlating with clinical findings [87].

One of the basic problems of this analysis was, that the structural description of the brain surface has been derived from a single 2D view of the rendered data as illustrated in figure 6.8. The red lines indicate the ridges detected by applying geometry-limited diffusion on the 2D view of the 3D rendering of the brain surface. They have been used to describe and analyze the sulco-gyral pattern of the brains under study. This way the essentially 3D structure of the cortex has been coded and analyzed by view-dependent 2D descriptors setting serious limits for the subsequent analysis. The usage of real 3D shape features for the description of the cortical structures has been defined as the major preliminary for more precise and reliable statistical analysis of the data.

Skeletal representation offers a promising way to generate more precise descriptors of the sulco-gyral foldings. As a first step we have used our skeletonization software to produce a skeletal description of the inner surface of the cortex. On the inner surface which is the transition between the cortex and the white brain matter the sulci and gyri are much more visible. This is not only due to partial volume effects which tend to blur out the narrow sulci when the cortex is segmented from MR images but also to the fact that the cortical gyri are pressed together and there is only a flymsy membrane between them. The result we are going to present here should be seen as a feasibility study which shows that our skeletonization method can be used to produce a moderately complex description of the cortical brain structure.

In figure 6.9 three slices in each of the three slice directions (sagittal, axial and coronal) through the original gradient echo MR dataset are shown. The red lines show the segmentation of the inner cortical surface. The data was provided by Prof. F. Jolesz and Prof. R. Kikinis from the MRI Division of the Brigham and Women's Hospital Boston. Together with the original MR data there is a labelling of the main brain structures available which has been produced.
Figure 6.8: 3D surface rendering of the brain surface from MR data. The red lines exhibit the locations of sulci. They have been produced by applying geometry-limited diffusion to the 2D image.

by neuroanatomists. This labeling has been used to segment the inner cortical surface and will be helpful in the evaluation of the skeletal structure, in particular to check how different neuroanatomical structures are represented in the skeleton. Before going ahead all holes in the segmentation have been filled out. This is an important step to get rid of some of the topological noise present in the segmentation. In fact even only one voxel sized holes in the segmentation would inevitably produce a sphere in the skeletal description whose radius depends not on the size of the hole but on the depth inside the object where it is located. Hole filling is easily done by a connected component labeling on the image background, extracting the largest component and taking the reverse image of it. What we can’t do is to get rid of small handles in the object due to topological noise. A remedy would be to use a segmentation method based on the deformation of a region with a priori known topological properties as proposed in [107].

A fusible stereo pair of a 3D rendering of the segmented data can be seen in figure 6.10. The sampling of the surface at voxel level resulted in 205,848 boundary points. With this amount of input points the Voronoi diagram generation becomes a bottleneck of the skeletonization process. An overview of the data contained in the Delaunay triangulation and in the Voronoi diagram is presented in the first row of table 6.4. Sagital views of the left side of the resulting skeletons after regularization based on the ridge-strength measure using several thresholds are shown in figure 6.11. They have been coloured according to the ridge-strength measure. The stepwise reduction of the Voronoi diagram can be well seen on these images and also from the numerical data in table 6.4. By this image sequence it becomes apparent that the gradual regularization of the Voronoi diagram already allows us to recognize some of the most important cortical gyri and sulci (which are not really represented by the inner skeleton but are located in between the skeletal sheets representing gyri). They are indicated by the arrows. However it becomes also evident that many cortical gyri are represented by Voronoi faces having a rather weak ridge-strength measure. We will have to account for that in the postprocessing. Finally the topological skeleton shows that there exist many handles in the segmentation.

After regularization the resulting skeletal structure is still very complex and cannot be directly
6.3 Analysis of the cortical brain structure

Figure 6.9: Three sections in each of the three slice directions (sagittal a c, axial d f and coronal g i) of the original gradient echo MR data set. The image size is $256 \times 256 \times 124$ and the shown slices belong to the following slice positions: 89 (a), 127 (b), 156 (c), 81 (d), 116 (e), 152 (f), 28 (g), 58 (h) and 77 (i). The red lines indicate the segmentation of the inner cortical surface.

used for the generation of a 3D shape descriptor. The postprocessing described in chapter 5 should lead to the extraction of a few dominant skeletal sheets. To account for the weak discrimination of the ridge-strength measure the postprocessing is applied iteratively. The working steps can be summarized as follows:

1. Apply a strong regularization on the Voronoi diagram. We used a threshold of about 25.

2. Perform a grouping on the remaining Voronoi faces and identify the Voronoi faces which represent dominant unbranched skeletal sheets by eventually joining several groups together or cutting others in two.

3. Relax the regularization by setting the threshold to a lower value.
Figure 6.10: A 3D rendering of the inner surface of the cortex of a human brain, shown as fusible stereo pairs.

<table>
<thead>
<tr>
<th>regularization threshold</th>
<th>Delaunay triangulation</th>
<th>Voronoi diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vertices</td>
<td>edges</td>
</tr>
<tr>
<td>—</td>
<td>205'848</td>
<td>791'652</td>
</tr>
<tr>
<td>5</td>
<td>229'590</td>
<td>495'437</td>
</tr>
<tr>
<td>10</td>
<td>171'958</td>
<td>327'368</td>
</tr>
<tr>
<td>15</td>
<td>116'844</td>
<td>206'712</td>
</tr>
<tr>
<td>20</td>
<td>72'065</td>
<td>116'169</td>
</tr>
<tr>
<td>25</td>
<td>15'178</td>
<td>14'009</td>
</tr>
</tbody>
</table>

Table 6.4: Number of elements contained in the Delaunay triangulation and in the Voronoi diagram of the inner cortical surface. The reduction of the Voronoi diagram for different regularization thresholds is shown.

4. Perform again a grouping this time on the not yet identified Voronoi faces. Complete the previously identified skeletal sheets and identify new ones.

5. Go up again one level in the regularization and continue in the same manner.

The inner cortical surface can be easily divided into two parts, each of which corresponds to one of the two brain halves. The results given in the rest of the section have been computed only for the regularized Voronoi diagram lying inside the left brain halve. The only objective was to reduce the amount of data involved. Note that the data reduction has been made only after the computation of the Voronoi diagram and the regularization. So the skeleton is the same as the left part of the skeleton of the entire inner cortical surface.

The iteratively applied postprocessing lead to 81 face groups shown in figure 6.12. They represent the main unbranched skeletal sheets of the left inner cortical surface. Hence the initial description of the left skeleton with over 100'000 Voronoi faces has been compiled into a manageable set of less than 100 unbranched skeletal sheets. The skeletal sheets are running in the middle
6.3 Analysis of the cortical brain structure

Figure 6.11: The skeleton of the inner cortical surface at different levels of regularization coloured according to the ridge-strength measure which has been used for the regularization. The threshold has been set to the following values: 0 (a) i.e. no regularization at all, 10 (b), 15 (c), 20 (d), 25 (e), 30 (f), 35 (g), 40 (h) and 45 (i) i.e. maximal regularization which leads to the topological skeleton.

of the more pronounced gyri as one can see from the superposition of a part of the original object with its skeleton in figure 6.13. The gyration pattern of the cortical surface is now coded in the branching of these skeletal sheets. Current research is concentrating onto the extraction and quantitative description of the branching lines between neighbouring skeletal sheets, leading to truly 3D descriptors of the sulco-gyral structure of cortical regions like e.g. the temporal lobes.

It remains the question (which only can be addressed by neuroanatomical experts) whether these sheets should be subdivided again in order to conform the skeletal shape descriptor to the cortical topography so that each sheet possibly represents one structural part of the left inner
cortical surface.

Once we have obtained the skeletal descriptor of the left inner cortical surface we can use it to reconstruct the original object labeling each part of the reconstruction according to the skeletal sheet from which it has been reconstructed. This gives us an impression which structural parts of the surface are represented by which skeletal parts. Figure 6.14 shows such the reconstruction of the original shape using an increasing number of the most dominant skeletal sheets from figure 6.12. Obviously the reconstruction of the shape is not perfect unless we use much more skeletal sheets which however are not significant for the cortical structure. To give a feeling of how good the chosen skeletal sheets represent the original shape we show the original shape coloured according to its distance from the reconstructions shown in figure 6.15. One can see that the only regions where the original shape is poorly reconstructed are those which have an axial local symmetry type. This is due to the fact that up to now we have defined the postprocessing steps only on Voronoi faces and not on Voronoi edges which represent axial symmetries.
Figure 6.12: Upper row: The 81 skeletal sheets of the left inner cortical surface shown as a fusible stereo pair. Different sheets have been assigned different colours. Lower row: The 20 most dominant skeletal sheets of the same object.
Figure 6.13: Superposition of the skeleton with a part of the original object. The intersection plane is shown in white and the rest of the original object in cyan. One can see the red skeletal sheets running in the 'middle' of the object.
Figure 6.14: The reconstruction of the left inner cortical surface using an increasing number of the most dominant skeletal sheets. The colours indicate which part of the shape has been reconstructed from which skeletal sheet. The number of skeletal sheets used is: 3 (a), 6 (b), 15 (c), all (d).
Figure 6.15: Difference in voxels between the left inner cortical surface and its reconstruction from the most dominant skeletal sheets. The number of skeletal sheets used is: 3 (a), 6 (b), 15 (c), all (d).
Chapter 7

Conclusions

7.1 Summary

Shape description is undoubtedly a major issue in computer vision. Fundamentally there are two different ways to describe shape: one using the boundary between the shape and its background, the other based on the region occupied by the shape. The resulting descriptors are basically equivalent offering a dual representation of shape. While boundary-based methods are well established and widely used much less attention has been paid to the region-based approach in the past. This is mostly due to the simplicity and effectiveness of boundary-based methods in 2D. However with the ever growing need of handling three-dimensional shape and due to the complexity of shape description in 3D for both boundary-based as well as region-based methods, this imbalance is attenuated. Although this thesis focus on region-based shape description I do not judge one approach superior to the other but I would rather argue that the two approaches should be used to complement each other. In such a framework the strength of region-based shape descriptors lies in their insensitivity to geometrical noise on the boundary, their ability to capture global shape features and their potential to reveal the hierarchical organization of shape.

In the introduction I suggested that the choice of the appropriate shape description method should be guided by the type of shape to be described and by the task to be solved in the higher levels of the vision process. The type of shape concerned in this thesis is the one of biological objects. It has already been pointed out by D'Arcy Thompson [46] that living objects are the result of axial growth processes. Later Blum [21] proposed the concept of local object symmetry to mathematically capture axial growth. The local object symmetries are generated by a process called the symmetry axis transform (SAT). The representation of the object that comes out is often called the skeleton of the object and accordingly the term skeletonization is used as a synonym for the SAT. In chapter 1 the conceptual aspects of the SAT have been presented. The following important facts have emerged

1. The SAT is suited to represent axial and mirror symmetry but poorly represents rotational symmetry.

2. The SAT is a symmetry selection process. The advantage thereof is that the skeleton consists of a manageable set of local symmetries. The drawback is the notorious sensitivity of skeletons to topological noise.

3. Not all parts of the skeleton represent the same amount of symmetry information. Some parts are just due to the fact that skeletons are homotopically equivalent to the object they represent. These parts make up the connection between the proper local symmetry axes. They play a decisive role in the hierarchical organization of the symmetry axes.

4. The SAT essentially produces a skeletal part for each 'convex' part of the shape regardless of its relative size with respect to the overall shape. This makes skeletonization extremely unstable unless a subsequent regularization which removes spurious skeletal parts is added.
The important consequence is that one cannot speak about stability of the skeleton independently from the regularization.

The SAT is defined in the continuum and strategies had to be developed to implement it on the discrete image raster. In chapter 2 I have given an overview of the different skeletonization techniques that have emerged in the literature. The two standard types of implementations of the SAT on the image raster are the detection of distance ridges and thinning procedures. Both of them are not able to produce 'topologically' as well as 'geometrically' correct skeletons. There are some analytical methods which do not have these problems. However they are not applicable to general biological shape but only to special classes of shape. As far as I know, the only approach that succeeds in simultaneously fulfilling the two requirements are the Voronoi skeletons which represent a semi-continuous implementation of the SAT. At the time when this work initiated Voronoi skeletons have only been investigated in 2D. So the primary goal of this thesis has been to extend this promising method to 3D. Thereby a major undertaking was to make the method applicable for large shapes described by several hundred thousand boundary points.

In chapter 3 I have shown that the commonly used discrete representation of continuous shapes with voxels is not adequate as soon as methods need consistent definitions of topological notions as boundary, connectivity etc. A presentation of the basic concept of cell complexes as proposed by Kovalevsky for the use in image processing is given. An important contribution is the proof that, with a uniform sampling scheme which takes a sample at the center point of every voxel face on the boundary of the shape, the inner Voronoi diagram is homotopically equivalent to the shape. An earlier proof given by Brandt and Algazi [30] has been shown to fail for this sampling scheme contrary to their findings. Next I have given an overview of the state of the art in computing Voronoi diagrams in 3D. Thereby existing algorithms had to be extended to comply with the following requirements posed by the envisaged task

1. The algorithm must be able to deal correctly with cospherical points since the sample points are taken from the image raster which is a regular grid.

2. The algorithm must be able to deal with very large pointsets. Hence the economical use of memory and the speed of the algorithm become crucial entities.

In chapter 4 the regularization of the Voronoi diagram is studied. The task of the regularization is to reduce the Voronoi diagram to its stable part which is the Voronoi skeleton. It is an iterative process which needs a mean to guarantee at every moment that the reduced Voronoi diagram is homotopically equivalent to the initial one. I have developed such homotopy checks based on local information of the Voronoi diagram or equivalently the Delaunay triangulation. I have presented homotopy checks for both the Voronoi faces and the Delaunay polyhedra. The second essential part of the regularization is the definition of importance measures expressing the significance of a skeletal part (corresponding to a part of the Voronoi diagram) relative to the overall shape. I have shown that a distinction between global and local measures can be made. While local measures basically make explicit the 'amount' of local object symmetry a skeletal part represents, global ones additionally take into account what part of the overall shape depends on a skeletal part. It has been shown how global measures can be used to reveal the hierarchical organization of shape in 2D. Presently there is no way in sight to define such global measures for 3D shape. As a consequence of this lack the hierarchic organization of 3D form cannot be made explicit by automatic procedures. I have shown by a careful analysis of the topological structure of the Voronoi diagrams that the fundamental problem lies in the cyclic structure of the Voronoi graph. The Voronoi edges for instance form a net-like graph on which no natural ordering can be defined. The problem was already present in 2D but since the cycles in the 2D Voronoi graph corresponded to the topological structure of the shape it has not been noticed. Hence in 3D the regularization is based on local importance measures. One problem of local measures is that they tend to be weak near the boundary of the object. That's why other authors supply their local measures with an additional condition depending on the element's depth in the object. I presented a local importance measure called ridge strength measure which is derived from a
7.2 Outlook

In this section I would like to point out the directions in which according to my opinion future research on 3D Voronoi skeletons should go. These are

- Generation of the Voronoi skeleton
- Postprocessing
- Applications

Generation of the Voronoi skeleton

- In order to account for the topological sensitivity of the skeletonization one could use homotopically deformable regions to perform the segmentation as proposed by Malandain, Bertrand and Ayache [107]. The prerequisite is that the topological properties of the object to be segmented are a priori known.
- The boundary of the object could be given with subpixel accuracy by many segmentation routines. Sampling the object’s boundary at subpixel level would avoid the generation of many Voronoi elements which are due to the discretization.
- The representation of the object by voxels limits our knowledge about the original object and introduces an approximation error in the range of the voxel size. It would be interesting

Euclidean distance map of the object and combines in an elegant way the 'amount' of local symmetry carried by an element with its depth in the object. Finally I propose a scheme to automatically select the type of local symmetry of an object. For that purpose the Euclidean distance map of an object is analyzed with the help of the Hessian matrix. The eigenvalues of the Hessian matrix at a point in space can be used to discriminate between rotational, axial and mirror symmetry.

Most authors are already content with the capability to generate 'skeleton pictures', i.e. with the set of pixels or voxels that belong to the object's skeleton. However for the practical applicability of skeletons a graph-like description is needed. The Voronoi skeletons have the advantage that they inherently provide such a graph structure. But the microscopic sampling of the object's boundary with points is reflected by the huge amount of microscopic Voronoi faces making up the object’s skeleton. In chapter 5 I have demonstrated that these numerous microscopic Voronoi faces can be turned into a manageable set of macroscopic unbranched skeletal sheets. Due to the lack of global importance measures the shape hierarchy cannot be automatically detected. Nevertheless a shape hierarchy can often be manually established by the user. For that purpose I have developed an interactive tool-box which assists the user in performing this task and given some ideas to further improve these tools in the future.

The last chapter has demonstrated that in spite of all theoretical problems the explicit goal of this thesis to generate 3D Voronoi skeletons of highly complex biological shape has been reached. Already at this early stage Voronoi skeletons have given us a new and exciting view of the complex structure of the human brain's cortical surface. I expect that in collaboration with clinical experts Voronoi skeletons can provide a powerful tool to gain new insights into and to characterize biological shape. However the most promising way to use skeletons is in combination with the full set of local symmetries. In such a framework skeletons are computed just once for a representative model of the shape which is to be recognized in a scene or which should be matched to another instance of the same type of shape. The unstructured set of all local symmetries is derived directly from the unsegmented target scene. The skeleton of the model can then be found in this full set of local symmetries.

Finally I would like to note that skeletons may find just as much interest as in computer vision in other fields like computer graphics (e.g. to model the motion and deformation of complex objects) or robotics (e.g. for time-critical collision detection as in [82]).
to study the influence of this uncertainty in the position of the boundary points on the position of the approximate skeleton.

- The determination of the local symmetry type by characterizing the distance map with the Hessian matrix should be integrated into the regularization.

- Up to now only local regularization measures exist in 3D and I don't see how global measures could be mathematically defined. On the other hand there might be some heuristics leading to the definition of global measures. Anyway, I think that the search for global measures should not be done on the 'microscopic' elements of the Voronoi diagram but on the 'macroscopic' skeletal sheets resulting from a first regularization of the Voronoi diagram with a local measure and a grouping of the remaining Voronoi elements. In order to define a global measure based on skeletal sheets one should use the adjacency of these sheets to accumulate some local measure already defined for each skeletal sheet. To measure the local importance of a skeletal sheet one could use for instance the volume of the shape represented by the sheet.

- The regularization has always been done proceeding from the outside to the inside. The observation that the skeleton of an object is usually more complex near the boundary than deep inside the object could motivate another approach to define the regularization.
  - Select the topological skeleton as a basis to extract the skeleton from the Voronoi diagram. We have seen that the topological skeleton is not unique. Hence additional conditions must be given to select a unique part of the Voronoi diagram as basic skeleton. I would suggest to select the parts which lie deepest in the object.
  - Extend this basic skeleton by incorporating the most prominent adjacent parts of the Voronoi diagram until the boundary is reached. The difficulty here lies in the determination of the most prominent part. The local importance measures are not discriminating enough to decide reliably which part should be selected. So one could also use another criterion, e.g. to choose the part which best can guarantee smooth continuation of the skeleton.
  - Then one could use the reconstruction algorithm to identify other prominent parts of the shape which are not yet represented in the skeleton and the skeleton could be extended to incorporate these parts, too.
  - This process can be iteratively repeated until the shape can be sufficiently well reconstructed from the skeleton.

The basic problem with this approach is the difficulty to define checks for homotopical equivalence similar as for the processing from the outside to the inside.

Postprocessing

- The face groups resulting from the automatic grouping process often must be further aggregated manually into large unbranched skeletal sheets. To further automate this process the ideas described in section 5.3 should be investigated. Moreover the interactive tools developed to assist the user in the manual aggregation should be extended in the following directions
  - provide an easy definition of outlines to subdivide face groups that have become too large by the automatic grouping process.
  - supply additional information of the face group to the user; e.g. the part of the shape represented by the group.

- The postprocessing should be extended to the lineal skeletal parts.
• The representation of the skeletal sheets by groups of microscopic Voronoi faces is not yet satisfactory. The skeletal sheets should be approximated for example by splines. A parametrization of the skeletal sheets is also necessary for the elastic matching.

• Similarly as proposed for the regularization, the shape hierarchy could be determined by processing from the inside to the outside. The topological skeleton or rather an extension of it stays at the first hierarchical level. The next levels are determined by looking at how much of the shape is reconstructed additionally by the adjacent skeletal sheets. Skeletal parts which reconstruct an additional part of similar size defined to be on the same hierarchical level.

Applications

• The aggregation of skeletal sheets should be revised in collaboration with neuroanatomical experts so that each sheet possibly represents one structural part of the cortical surface. Thereby the fact that the original dataset has already been labeled could be used to see whether the part which is reconstructed from a single sheet has in its neighborhood only labels of the same type.

• Up to now the Voronoi skeleton has been computed only on the cortical surface of a single human brain. It would be interesting to compare the Voronoi skeleton of the cortical surface of a set of different patients. In this way the supposition that the skeletal parts which represent the large scale features of the cortical surface are roughly the same for different individuals could be validated.

• Another interesting study would be the comparison of the skeletons of the two brain halves. Can the asymmetry between the two brain halves be assessed by projecting corresponding parts of the two skeletons onto each other and measuring the amount of non-corresponding parts?

• The next question that should be addressed is the elastic mapping of the skeleton of a model into the full symmetry set. In a first step one should try to find the skeleton in the full symmetry set of the model itself. Afterwards one could try to elastically map the skeletal parts which represent the most prominent features of the shape into the full symmetry set of a similar object.
Appendix A

Voronoi diagram

A.1 Proofs of fundamental Voronoi diagram properties

We cite here the proofs given by Tanemura et al. [179] of the properties P2-P4 which are used by our implementation of computing the Delaunay triangulation of 3D pointsets.

First two lemmas used for the proofs are introduced:

L1 Let $C$ be the intersection circle of two spheres $C_0$ and $C_1$. Take a point $p$ outside $C_0$ but inside $C_1$. Let $C_2$ be the sphere defined by the circle $C$ and the point $p$. Then the center $c_2$ of sphere $C_2$ lies on the segment $c_0c_1$.

Figure A.1: Illustration of proof of lemma 1

PROOF. The center $c_2$ must have the same distance to all points on the circle $C$. The locus of points having the same distance from all points of the circle $C$ is the line orthogonal to the circle and passing through its center $c$. Hence $c_2$ must lie on the line passing through $c_0$ and $c_1$. In the following we will show that it lies between $c_0$ and $c_1$. Consider the plane defined by the centers $c_0, c_1, c_2$ and the point $p$ (see left part of figure A.1). The points $p_a$ and $p_b$ are the intersection of the plane with the circle $C$. The segment $p_ap_b$ is a diameter of circle $C$. Let $p_0$ and $p_1$ be the intersection points of the line $p_0p$ with the spheres $C_0$ and $C_1$.

\[
\gamma = 360 - (\gamma_0 + \gamma_1)
\]
\[
2(\alpha+\beta) = 360 - (\gamma + \gamma_1)
\]

Figure A.1: Illustration of proof of lemma 1
and $C_1$, respectively. Since $p$ is outside $C_0$ but inside $C_1$ it follows that

$$\langle p_a p_0 p_b \rangle < \langle p_a p_1 p_b \rangle$$

On the right side of figure A.1 we see that the angle at the center of the circle above the arc $p_a p_b$ is twice the angle at a point on the circle above the same arc $p_a p_b$. Hence

$$\langle p_a c_0 p_b \rangle = 2 \langle p_a p_0 p_b \rangle, \quad \langle p_a c_2 p_b \rangle = 2 \langle p_a p_1 p_b \rangle, \quad \langle p_a c_1 p_b \rangle = 2 \langle p_a p_1 p_b \rangle$$

and therefore

$$\langle p_a c_0 p_b \rangle < \langle p_a c_2 p_b \rangle < \langle p_a c_1 p_b \rangle$$

i.e. $c_2$ lies between $c_0$ and $c_1$.

L.2 Assume the same configuration as in lemma 1 with the additional condition that the intersection circle $C$ between the two spheres $C_0$ and $C_1$ is a great circle of $C_0$ (i.e. contains the center $c_0$). Then the radius $r_2$ of sphere $C_2$ is smaller than the radius $r_1$ of sphere $C_1$.

PROOF. We have previously seen that $\langle p_a c_1 p_b \rangle$ is the maximum of the three angles above the same arc $p_a p_b$ and at one of the circle's centers. The additional condition means that this angle is 180 degrees and therefore all centers $c_0, c_1, c_2$ lie in the halfspace of the segment $p_a p_b$ which contains $p$. Hence to conclude that the radius $r_2$ is smaller than $r_1$ we simply compare the two isosceles triangles $\Delta(p_a c_2 p_b)$ and $\Delta(p_a c_1 p_b)$.

With lemma 1 and 2 Tanemura et al. give the following proofs of property P2-P4. For clarity the properties are formulated once again

P.2 Consider a set of triangles (or polygons in the case of cospherical sites) $\{s_0, s_1, s_j\}$ where $s_1$ is a nearest neighbor of $s_0$. Among them suppose a triangle $\{s_0, s_1, s_2\}$ has the minimum circumradius. Then the triangle is a Delaunay face.

PROOF. Let $C_1$ be the sphere whose diameter is the segment $s_0 s_1$. $C_1$ contains no other point since $s_1$ is a nearest neighbor of $s_0$. Let $C$ be the circumcircle of triangle $\{s_0, s_1, s_2\}$ and $C_2$ the sphere of which $C$ is a great circle. The intersection of the two spheres $C_1$ and $C_2$ is a great circle $C'$ of the sphere $C_1$ since the points $s_0$ and $s_1$ are intersection points of the two spheres. Then lemma 2 can be applied to conclude that every site $s_j$ lying outside $C_1$ but inside $C_2$ would form a triangle $\{s_0, s_1, s_j\}$ which is not allowed by the condition that the triangle $\{s_0, s_1, s_2\}$ must have the minimum circumradius. Hence the sphere $C_2$ contains no sites. Then the center $c_2$ which is equidistant to all sites $\{s_0, s_1, s_j\}$ is also closer to each one of the sites $\{s_0, s_1, s_j\}$ than to any other site. Hence the Voronoi cell $V(R)$ of $R = \{s_0, s_1, s_j\}$ is non-empty and therefore the triangle $\{s_0, s_1, s_2\}$ is a Delaunay triangle.

P.3 Consider a set of tetrahedra (or polyhedra in the case of cospherical sites) $\{s_0, s_1, s_2, s_j\}$. Among them suppose a tetrahedron $\{s_0, s_1, s_2, s_3\}$ has the minimum circumradius. Then the tetrahedron is a Delaunay tetrahedron.

PROOF. Let $C_3$ be the circumsphere of tetrahedron $\{s_0, s_1, s_2, s_3\}$. The intersection of the two spheres $C_2$ and $C_3$ is a great circle $C'$ of the sphere $C_2$ since the points $s_0, s_1$ and $s_2$ of the triangle are intersection points of both spheres. Property P2 says that the sphere $C_2$ contains no site and lemma 2 can be applied to conclude that every site $s_j$ lying outside $C_2$ but inside $C_3$ would form a tetrahedron $\{s_0, s_1, s_2, s_j\}$ with a smaller circumradius which is not allowed by the condition that the tetrahedron $\{s_0, s_1, s_2, s_3\}$ must have the minimum circumradius. Hence the circumsphere $C_3$ of tetrahedron $\{s_0, s_1, s_2, s_3\}$ contains no sites and thanks to property P0 we can conclude that it is a Delaunay tetrahedron.

P.4 Suppose the tetrahedron $\{s_0, s_1, s_2, s_3\}$ is a Delaunay tetrahedron. Consider a set of tetrahedra $\{s_0, s_1, s_2, s_j\}$ where $s_j$ is in the halfspace $H_f^{-s_3}$ of face $f = \{s_0, s_1, s_2\}$ which does not contain the site $s_3$. Among them suppose a tetrahedron $\{s_0, s_1, s_2, s_4\}$ has the minimum signed circumradius. The signed circumradius has the same absolute value as the
circumradius and its sign is negative if \( s_4 \) lies in the same halfspace as \( s_3 \). Then the tetrahedron is a Delaunay tetrahedron.

PROOF. Let \( C_3 \) be the circumsphere of tetrahedron \( \{s_0, s_1, s_2, s_3\} \) and \( C_4 \) the circumsphere of tetrahedron \( \{s_0, s_1, s_2, s_4\} \). Property P0 says that \( C_3 \) contains no sites. Assume a site \( s_j \) lies inside sphere \( C_4 \) and outside sphere \( C_3 \). Since the intersection of the two spheres is the circumcircle of the common face \( \{s_0, s_1, s_2\} \) lemma 1 can be applied, i.e. the center \( c_j \) of the circumsphere \( C_j \) of tetrahedron \( \{s_0, s_1, s_2, s_j\} \) would lie between the center \( c_3 \) and \( c_4 \). From the drawing on the left of figure A.1 we can see that the radius of a sphere whose circumcenter lies on the perpendicular bisector of segment \( \overline{P_aP_b} \) is minimal if the sphere’s center is on the segment and monotonically grows when moving the sphere’s center on the bisector away from the segment. Hence the signed circumradii appear in increasing order on the bisector and since the signed circumradius of \( C_3 \) is negative, the circumradius of \( C_j \) is bigger than the one of \( C_3 \) but smaller than the one of \( C_4 \). This contradicts the condition that the tetrahedron \( \{s_0, s_1, s_2, s_4\} \) has the minimum signed circumradius. Hence the circumsphere \( C_4 \) contains no site and the tetrahedron \( \{s_0, s_1, s_2, s_4\} \) is a Delaunay tetrahedron.
Appendix B

Boundary sampling

B.1 Sufficient sampling density for r-regular shapes

The skeleton of a shape can be approximated by the inner Voronoi diagram of a discrete sampling of the object's boundary. Schmitt [157] showed that this approximation converges to the actual skeleton of the object if the sampling density becomes infinite. However, for a practical implementation one is interested into a finite sampling density for reliable skeleton approximation. Brandt gives three conditions for a reliable skeleton approximation.

1. The sample points lie on the boundary of the shape regenerated from the shape's approximate skeleton.
2. The approximate skeleton is homotopically equivalent to the original shape.
3. The approximate skeleton converges to the exact skeleton if the sampling density becomes infinite.

He shows that these conditions can be fulfilled for a given sampling density if the continuous shape underlying the discrete 'voxel object' is assumed to be r-regular with a certain radius r of r-regularity. Let us give first some necessary definitions.

For a shape $O$ we consider a sampling $W_n = \{w_i\} \subset \partial(O)$ consisting of $n$ sampling points. The sampling density of $W_n$ is then defined as

$$\omega_n = \max_{p \in \partial(O)} \delta(p, W_n)$$

which is the maximal distance between a boundary point to its next sampling point.

A shape $O$ is called r-regular if it is morphologically open and closed with respect to a d-dimensional ball of radius $r > 0$:

$$O = (O \ominus B_r) \oplus B_r = (O \oplus B_r) \ominus B_r$$

An r-regular shape has the following important properties:

1. The curvature on the boundary of an r-regular shape is never greater than $\frac{1}{r}$.
2. The radius of a maximal ball inscribed to the shape is never smaller than $r$, i.e. the shape will never have a canal narrower than $2r$.

As a consequence of the second property Brandt shows that for a sampling density $\omega_n < 2r$ the boundary of the r-regular shape cuts each Voronoi region into two simply-connected regions, belonging respectively to the interior and to the exterior of the shape. With the help of this central property he shows that the resulting approximate skeleton satisfies the desired criteria.

$\ominus$ means morphological erosion and $\oplus$ morphological dilation.
It remains to find for all possible objects of the discrete image raster an underlying r-regular shape which fulfills \( \omega_n < 2r \) for the sampling defined by the intersection of the r-regular shape with the image raster. Brandt defined an r-regular shape for 2D objects by replacing each corner of the shape boundary with a quarter circle of radius \( \frac{h}{2} \) where \( h \) is the pixel spacing. This definition cannot be directly generalized to 3D. Therefore we prefer to define an r-regular shape by mathematical morphology which covers both the 2D and the 3D case. For that purpose the object pixels/voxels are first replaced with circles/spheres of radius \( \frac{h}{2} \) around the pixel/voxel centerpoints and then a morphological closing with a circle/sphere of radius \( \frac{h}{2} \) is performed. The morphological closing operation is defined as a dilation followed by an erosion and the replacement of the object pixels/voxels by circles/spheres can be described by a dilation of the pixels/voxels centerpoints. Then the construction can be described as follows:

1. The object is described as a set of connected pixels/voxels. Extract their centerpoints in order to get a set of points \( P = \{ p_i \} \)
2. Compute the set \( P' = P \oplus B_{r_d} \)
3. Compute the r-regular shape \( O = P' \ominus B_{r_e} \)

The r-regularity of the resulting shape \( O \) depends on three parameters

- the largest distance \( d_{\text{max}} \) between the centerpoints of two adjacent pixels/voxels.
- the dilation radius \( r_d \) and
- the erosion radius \( r_e \)

![Figure B.1: r-regularity](image)

The parameters \( r_d, r_e \) and the distance \( d \) between two pixel/voxel centerpoints are related as shown in figure B.1. The parameters cannot be chosen arbitrarily in order to get an r-regular shape. Three different situations can arise as shown in figure B.2:
The radii of dilation and erosion and the distance between the centerpoints of adjacent pixels/voxels define whether the morphological operations lead to a connected r-regular shape (case a), disconnected shapes with discontinuities in the boundary curvature (case b) or to disconnected balls.

(a) the radii \( r_d \) of dilation and \( r_e \) of erosion are chosen in such a way that the resulting shape is connected. In this case the r-regularity is given by the following formula:

\[
(r_e + r)^2 = r_d^2 - \left( \frac{d_{\max}}{2} \right)^2
\]

(b) the radii are selected in such a way that after dilation the shape is connected but too narrow. Therefore the erosion disconnects it again and produces discontinuities in the curvature of the resulting shape.

(c) the radius \( r_d \) of dilation is smaller than half the distance \( d \). Hence already after dilation the shape is disconnected.

As one can see from figure B.2 each of these three situations shows up in a distinct interval of \( d \). The three intervals are given by the following inequalities:

(a)

\[
0 \leq \left( \frac{d}{2} \right)^2 \leq r_d^2 - r_e^2
\]
To complete the definition of an r-regular shape assigned to an object described in the pixel/voxel representation we have to find an appropriate value for the two parameters \( r_d \) and \( r_e \). The third parameter \( d \) is given by the discrete image raster. It can assume only a discrete number of values and we must consider only values which express the distance between two midpoints of neighbouring pixels/voxels. In 2D this distance is \( h \) or \( h \cdot \sqrt{2} \) where \( h \) designates the pixel spacing. In 3D it is \( h, h \cdot \sqrt{2} \) or \( h \cdot \sqrt{3} \). Then we have to chose \( r_d \) and \( r_e \) in such a way that the interval of \( d \) inside which situation (b) arises contains neither \( d = d_{\text{max}} \) nor \( d = 2 \cdot h \) which is the distance to the next non-adjacent pixel/voxel.

Hence we have found two inequalities which must be fulfilled by \( r_d \) and \( r_e \). But this is not enough since we want that the pixel/voxel representation of the shape and the underlying r-regular shape intersect at least at the sampling points which describe the boundary of the object. Since the sampling points are taken from the centers of the pixel sides in 2D and from the centers of the voxel faces in 3D this can be achieved if the difference between dilation and erosion radius equals half the pixel spacing. Hence for both the 2D and the 3D case we have now three conditions restricting the possible choice of \( r_d \) and \( r_e \):

1. \( \sqrt{r_d^2 - r_e^2} > \frac{h \cdot \sqrt{\text{dimension}}}{2} \)
2. \( r_d \leq h \)
3. \( r_d - r_e = \frac{h}{2} \)

This gives the following limits for \( r_d \) and \( r_e \) in the 2D case

\[
\begin{align*}
    h &\geq r_d > \frac{3}{4} h \\
    \frac{3}{4} h &\geq r_e > \frac{1}{4} h
\end{align*}
\]

and in the 3D case

\[
\begin{align*}
    h &\geq r_d > h \\
    \frac{3}{4} h &\geq r_e > \frac{1}{4} h
\end{align*}
\]

Hence if we chose in the 2D case \( r_d = h \) and \( r_e = \frac{1}{2} h \) the underlying shape will be r-regular with \( r = \sqrt{r^2 - \frac{1}{2}} - r_e = \sqrt{h^2 - \frac{1}{2}} - \frac{1}{2} h \). Assuming that we have a pixel spacing of unit length \( (h = 1) \), the r-regularity is \( \frac{1}{2} \frac{\sqrt{2}}{2} = 0.207 \). Note that this is the same r-regularity as Brandt has found for his construction. Next we have to define a sampling scheme which gives a sampling \( W_n \) of the object and has density \( \omega_n < 2r \approx 0.414 \). In their paper Brandt and Algazi proposed to sample the r-regular shape at its intersection points with the image raster. They defined \( \omega_n \) on the left example given in figure B.3. In this special case \( \omega_n \) is the chord length of an eighth circle with radius \( \frac{1}{2} \), i.e. \( \omega_n = \sin(\frac{\pi}{8}) \approx 0.38 < 2r \) and the sampling is sufficiently dense. They state that this example is the worst case in 2D. However the right example of figure B.3 has adjacent sampling points which are more distant from each other which is reflected by a higher value of \( \omega_n = 0.5 > 2r \). Hence the sampling is not sufficient. Brandt didn't realize that his attempt to prove the sufficiency of uniform sampling at pixel level by the theory of regular sets fails! Note that the situation is equivalent when the r-regular shape is constructed morphologically. In the figure B.3 the resulting r-regular shape is drawn with dashed lines. The distance of a point on the curve to its nearest sample point is maximal for point \( P \), namely \( \delta(P, Q) \approx 0.517 \).
The problem could be solved by taking the samples at the intersection of the boundary of the r-regular shape with a grid which is twice as fine as the image raster. Then $\omega_n \approx 0.25 < 2r$ for the r-regular shapes defined by Brandt and $\omega_n \approx 0.259 < 2r$ for the r-regular shape obtained with morphology. However, one clearly wants to avoid a doubled sampling rate and our experience tells us that it is not necessary. The following observation could be a starting point to find a proof that the sampling scheme proposed by Brandt and Algazi is sufficient: The sampling density is not uniform all over the object but varies according to the pixel configurations. The maximal distance between two samples occurs e.g. in the case when two pixels at the object's border are direct neighbors (cf. figure B.3). But in that region the object is at least one pixel wide and the local r-regularity of the shape is higher. Hence the local sampling density should be defined and then related to the local r-regularity of the object shape. We do not continue in this direction since we are interested in the 3D case where another strategy has to be adopted anyway.

In 3D the only possible choice for the radii is $r_e = \frac{1}{2}h$ and $r_d = h$. However in this case the r-regularity of the resulting shape is 0! The problematic configuration is shown on the left side
Figure B.5: Determination of the sampling density $w_n$ in 3D

of figure B.4. On the right side a possible solution is given. The sampling density is doubled and additional voxels are inserted around the point where the object was only connected over a corner. Note that the accuracy of the information on the object's shape obtained during image acquisition is within the original voxel level, i.e. the actual boundary of the original object can differ by up to half the length of an original voxel. Hence if the sampling density used to extract the boundary points on the original voxel representation of the object is higher than the sampling density that has been used in the image acquisition to produce the object’s voxel representation, we are free to change this configuration in the limits of half an original voxel as we did. The worst case constellation which determines the r-regularity of the shape is then the one where two voxels are connected only over an edge. The distance of the two voxel centers is then $d_{\text{max}} = \sqrt{2} \times \frac{h}{2}$. Note that the original voxel spacing $h$ has been halved since the sampling is doubled. The r-regularity that comes out is $r = \frac{\sqrt{2}-1}{4} \approx 0.104$ for an original voxel spacing of $h = 1$.

The worst case on which the sampling density $w_n$ can be determined is shown in figure B.5. We adopt the same sampling scheme as in 2D, i.e. the grid lines of the image raster with voxels of length $\frac{h}{4}$ are intersected with the r-regular shape to determine the sampling points. Then $P_1$ is a point on the boundary of the r-regular shape which has maximal distance to its nearest sampling point $Q$. The distance can be determined from the figure; it is twice the chord length of an eighth circle with radius $r_e = \frac{h}{4}$. Hence assuming again that $h = 1$ we get $w_n = 2 \times 2r_e \times \sin(22.5^\circ) = \sin(22.5^\circ) \approx 0.383 > 2r$. This means that in 3D the proof does not even hold for the doubled sampling rate. However it does for the quadrupled sampling density. The point on the shape’s boundary which has maximal distance to its nearest sampling point is then point $P_0$ and its distance to $P_1$ becomes $\delta(P_0, P_1) = w_n = 0.5 \times \sin(22.5^\circ) \approx 0.191$ for which $w_n < 2r$ holds.

In summary the theory of r-regular shapes can only guarantee that the Voronoi diagram always remains inside the object if the sampling points are determined from the intersection of its voxel representation with a grid four times denser than the image raster.
B.2 Sufficiency of the boundary sampling

The sampling scheme we use throughout this work is defined on the voxel representation of the objects. A boundary point is generated at the center point of each voxel face which separates the object from its background. We have shown that this boundary sampling is sufficient to guarantee the homotopical equivalence of the object and the 'inner' Voronoi diagram of these boundary points if the following condition is fulfilled:

The 3-dimensional Voronoi diagram of the sampling intersected with the object's boundary must be identical to the 2-dimensional Voronoi diagram of the same sampling but computed on the 2-dimensional object boundary.

It is trivial to see that the 2-dimensional Voronoi diagram of the sampling points computed on the object's boundary consists of the voxel edges and voxel vertices which are lying on the object's boundary. Hence the voxel faces on the object's boundary form the 2-dimensional Voronoi regions of the boundary points.

![Figure B.6](image)

Figure B.6: The Voronoi diagram of the center points of all voxel faces is composed of octahedral Voronoi regions. The image shows one such region of which one boundary face has been removed. Inside the region one sees the voxel face and it's center point.

Next consider the 3-dimensional Voronoi region associated with a boundary point. It's intersection with the object's boundary is a 2-dimensional region. This region contains at least the interior of the voxel face at whose center the boundary point lies. This becomes obvious if one looks at the Voronoi diagram produced by the set of points which contains the center points of all voxel faces in the image raster. The structure of this Voronoi diagram is fairly simple. The Voronoi region associated with each center point is an octahedron formed by the four vertices of the voxel face on which the center point lies and the two midpoints of the two neighboring voxels. One such octahedron is shown in Figure B.6. One can see that the entire voxel face is contained in the Voronoi region of it's center point. Since the actual set of boundary points describing an object is a subset of the previously considered point set, the Voronoi regions can only grow and not shrink. Hence the interior of the voxel face at whose center a boundary point lies is contained in the Voronoi region of that boundary point. At the same time the object's boundary is completely described by a set of voxel faces and due to our sampling scheme we take a sampling point at the center of each of these faces. Hence the 2-dimensional region cannot contain additional points lying inside another voxel face.

What remains to consider are the edges and vertices which delimit the voxel faces. To prove that the intersection of the 3-dimensional Voronoi regions with the object's boundary is identical to the 2-dimensional Voronoi diagram of the sampling points we must show that these edges and vertices can only belong to the Voronoi regions of the boundary points lying on adjacent voxel
faces. From the previous considerations it is clear that they belong to the boundary of the Voronoi region associated with the center point of the enclosed voxel face. Therefore we exactly know the distance at which the boundary points of the neighboring Voronoi regions must be located. As consequence one immediately sees that for a voxel edge the only possible neighboring sampling points are the four center points of the voxel faces adjacent to the edge. Analogously for a voxel vertex the only possible neighboring sampling points are the twelve center points of the voxel faces adjacent to the vertex. This completes the proof.
Bibliography


Acknowledgements

First of all I would like to thank Dr. Gabor Székely for the fruitful collaboration during the past three years. We had many important discussions in which he gave me all his support to face the many technical and theoretical problems of this work. I appreciated also his careful reading of this thesis.

A special thank goes to Prof. Dr. Olaf Kübler who encouraged me to make a dissertation and provided me with an excellent research environment, not only from a technical point of view but also by creating a friendly and stimulating atmosphere in the lab.

I'm very grateful to Prof. Dr. Leila De Floriani for having accepted to act as co-examiner and for her hints especially regarding the parts of this thesis which are more related to computational geometry and discrete topology.

Last but not least I would like to thank all members of the lab for the many helpful hints and the good working climate. Especially I thank our secretary Vreni Vogt who contributed much to this atmosphere with her cheerful nature and helpfulness.
Curriculum Vitae

Name         Markus Olivier Näf
Date of birth 12.12.1965
citizen of    St. Gallen, Erlen(TG), Switzerland

Education

1972 – 1978   Primary school, Adliswil – Zürich
1978 – 1984   Gymnasium Zürich
              Maturität Typ B
1984 – 1985   Obligatory military service
1988 – 1989   One-year study visit in Italy to improve language skills
1985 – 1992   ETH Zürich, studies in Computer Science
              Diploma in the spring of 1992
1992 – 1993   Cultural stay in Florence with courses in theology and sociology
1996         PhD in Technical Sciences

Professional positions

1991         part-time employment as software-engineer at Spectrospin
1994 – 1996  research assistant at the Image Science Laboratory,
              ETH Zürich, director Prof. Dr. O. Kühler