Reconstruction of Surfaces from Unorganized Three–Dimensional Point Clouds

Dissertation
zur Erlangung des Grades des
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik
von

Dipl.-Inform. Robert Mencl

Lehrstuhl VII – Graphische Systeme
Fachbereich Informatik
Universität Dortmund

Dortmund
2001

Robert Mencl
Lehrstuhl VII – Graphische Systeme
Fachbereich Informatik
Universität Dortmund
D-44221 Dortmund

Tag der mündlichen Prüfung: 21. November 2001

Dekan: Prof. Dr. Thomas Herrmann

Gutachter:
Prof. Dr. Heinrich Müller
PD Dr. Paul Fischer

Contents

Part I

Introduction

Chapter 1

Introduction

This chapter defines the problem treated in this thesis: the surface (re-)construction problem, and illuminates different aspects of this problem. Then the contributions of the thesis are summarized. Finally an overview of the organization of the thesis is given.

1.1 The Problem

The problem treated in this thesis is

Surfaces from scattered point data:

Input: A set P of points in space which are sampled from a surface.

Output: A surface M so that the points of P lie on or close to M.

There is a wide range of applications for which surface construction from scattered point data is important. In particular, scanning of 3D shapes reaching from bodies to landscapes, directly accessible or not, with tactile, optical, ultrasonic, tomographic, and other sensors, is a rich source of data for the problem. The construction of surfaces is necessary because many methods and systems require surface data for further processing. Surfaces also open the application of the wide-spread surface-oriented visualization and rendering techniques. For example, surfaces may be used for visualizing other information e.g. coded in textures (data textures or real textures) mapped on the surface.

The given formulation of the surface construction problem is not very precise and lets many degrees of freedom of interpretation. From an application-based point of view, two *categories of tasks* can be distinguished: data analysis and surface reconstruction. *Data analysis* means that nothing is known about the surface from which the data originate. The task is to find the most reasonable solutions among usually several or even many possibilities. *Surface reconstruction* means that the surface from which the data are sampled is known, say in form of a real model, and the goal is to get a computer-based description of exactly this surface. This knowledge may be used in the selection of a favorable algorithm.

A proper reconstruction of the desired surface in the latter case can only be expected if it is sufficiently sampled. Sufficiency depends on the particular method of surface reconstruction. It might be formulated as a *sampling theorem* which should give sufficient conditions that can be easily checked. This aspect was neglected in research up to now, and little is known for most existing reconstruction algorithms on this aspect.

If surfaces are improperly sampled, a reconstruction method may cause artifacts which have to be dealt with. Like in classical sampling theory, pre-filtering e.g. in the sense of low-pass filtering may help to reduce artifacts at the costs of loss of details. Another possibility is interactive correction by the user which may be helpful if artifacts occur at some few isolated locations.

The opposite of insufficient sampling is that the sample data are unnecessarily dense. This happens in particular if a surface is sampled with uniform density. In that case the sample density required at fine details of the surface causes too many data points in regions of only minor variation. Several approaches to *data reduction* were proposed in literature [HDD+93]. We do not treat this topic here, but only give the hint that data reduction should consider the power of the reconstruction algorithm expressed in sampling theorems, a fact that also was not explicitly obeyed in the past.

The challenge of surface reconstruction is to find methods of reconstruction which cover a wide range of shapes, or, for a given area of application, to find a method of reconstruction which covers the shapes of this class reasonably. The challenge of data analysis is to find efficient enumeration algorithms yielding those of all feasible surfaces that come closest to the desired one. In particular, ways must be found to express which of the possible solutions are favorable.

The wide range of applications from which the data may emerge implies that the data can have quite different *properties* which may be considered at the solution of the surface interpolation problem. For example, the data may be sampled from surfaces that lie unique over a plane. In that case, a wide range of methods were developed which mainly focus on geometric properties like smoothness of the constructed surface [HL93].

Reconstruction may become more specific if the surface is captured in multiple samples (multiple view range images) that have to be fused. *Sample fusing* may need data transformation and fitting. We exclude these aspects from further discussion and refer e.g. to [TL94, CL96, SF97].

Sample data may contain *additional information on structure*. A typical example are tomographic data. In that case the points on a slice may be already connected by polygonal contour chains. Another example is that normal vectors are available at the data points. These additional informations may give additional hints on the unknown surface which may be considered in the construction algorithm. In particular, for interpolation or approximation of contour data, a variety of methods were developed [MK93]. In the following, no additional structural information is expected.

Finally, the *mathematical and data structural representation* of the derived surface has to be considered. The most common representation is the polygonal or triangular mesh representation. Because the representation by triangular meshes allows to express the topological properties of the surface, and because this is the most difficult sub-problem of surface construction, most known algorithms use this sort of representation. If higher smoothness than just continuity is required, either the parametric or the implicit surface representation may be used. Triangular meshes can be seen as a surface composed by parametrically represented linear surface patches. For surfaces of higher continuity patches of higher order are required. One way to obtain such surfaces is to start from a triangular mesh. For that reason, we have chosen the representation by triangular meshes for this thesis, and refer to literature for the problem of deriving smooth surfaces, for instance to [EH96, Guo97, FCGA97] in which smoothing of surfaces obtained from sample data is particularly emphasized.

1.2 The Contributions

In this thesis, a new surface reconstruction algorithm is presented which works well in practice, as has been demonstrated by application of an implementation to numerous data sets. Its particular features are

(1) reconstruction of open surfaces with boundaries of arbitrary genus as well as non-orientable surfaces,

(2) treatment of coarse sample sets of variable density,

(3) treatment of sharp edges, that is, locations of infinite curvature, like e.g. at the edges of a cube.

The algorithm can also be used for dense sample sets with a high number of points. Because of (2) efficiency can be gained by applying the algorithm to a sub-sample. This approach saves time and yields a lower number of triangles than for the original data.

We give formal arguments which explain why the algorithm works well. They consist of a rigorous definition of "reconstruction", and the demonstration of existence of sampling sets for which the algorithm is successful with high heuristic probability. This analysis focuses on compact closed surfaces with bounded curvature.

For the general case of surfaces with boundaries and sharp edges, we present heuristic arguments which contain potential for further work in direction of "provable" heuristics.

Further contributions are

(1) the application of the surface reconstruction algorithm for interactive shape design,

(2) a smoothing procedure for noise elimination in point clouds sampled from a surface.

The first contribution uses the property of the algorithm that already small point sets lead to reasonable surfaces. In this application, the sample points are used as control points for shape design. Additionally, the algorithm is capable of reconstructing surfaces out of arbitrary surface skeletons consisting of sets of surface edges between the sample points. As a result of these properties, the algorithm can be used very easily for user-defined locally restricted reconstructions with only little interaction.

The main advantage of the second contribution is that point smoothing is achieved without requiring a perfect interpolating surface.

The emphasis of this thesis lies on the reconstruction performance of the algorithm and on the demonstration of its practical performance, not on worst-case efficient algorithmic solution. Some hints on algorithmic aspects are given which have been shown useful during the implementation of the algorithm. For many computational subproblems worst-case efficient solutions are known in computational geometry.

1.3 Outline

The thesis is organized as follows.

In Chapter 2 an extensive survey of the state of the art of surface reconstruction algorithms is given. The existing algorithms are categorized according to their methodic approach, and advantages and disadvantages are discussed at the end of the chapter.

Chapter 3 introduces the basic structure of the algorithm, and explains in more detail than this chapter how the description of the algorithm and its analysis are organized.

The algorithm consists of two main phases, the construction of a skeleton graph or surface description graph, and the construction of a triangulation based on this graph. Chapters 4 and 5 are devoted to the first phase in that they present graph types well suited to the first phase.

Chapter 6 presents the definition of surface reconstruction and surface approximation on which the analysis of the algorithm is based. Chapter 7 analyzes the first phase of the algorithm.

The second phase, triangulation, is described in Chapter 8. It is analyzed in Chapter 9.

The application of the reconstruction algorithm to interactive geometric modeling is described in Chapter 10. Chapter 11 shows that the neighborhood information present in the developed graph types of Chapter 5 can be used in order to smooth noisy data sets.

Chapter 2

State of the Art

The surface construction problem has found considerable interest in the past, and is still an important topic of research. The purpose of this chapter is to find unifying basic methods common to independently developed solutions, coupled with a survey of existing algorithms. The identified basic classes are constructions based on spatial subdivision (Section 2.1), on distance functions (Section 2.2), on warping (Section 2.3), and on incremental surface growing (Section 2.4). In Section 2.5 the aspect is treated that an object represented in a sample data set may consist of several connected components. The survey closes with a discussion and categorization of our approach (Section 2.6).

2.1 Spatial Subdivision

Common to the approaches that can be characterized by "spatial subdivision" is that a bounding box of the set P of sample points is subdivided into disjoint cells. There is a variety of spatial decomposition techniques which were developed for different applications [LC87]. Typical examples are regular grids, adaptive schemes like octrees, or irregular schemes like tetrahedral meshes. Many of them can also be applied to surface construction.

The goal of construction algorithms based on spatial subdivision is to find cells related to the shape described by P. The cells can be selected in roughly two ways: surface–oriented and volume–oriented.

2.1.1 Surface–Oriented Cell Selection

The surface–oriented approach consists of the following basic steps.

Surface–oriented cell selection:

1. Decompose the space in cells.
2. Find those cells that are traversed by the surface.
3. Calculate a surface from the selected cells.

The Approach of Algorri and Schmitt

An example for surface–oriented cell selection is the algorithm of Algorri and Schmitt [AS96]. For the first step, the rectangular bounding box of the given data set is subdivided by a regular "voxel grid". "Voxel" stands for "volume element" and denotes a spatial cell of the grid.

In the second step, the algorithm extracts those voxels which are occupied by at least one point of the sample set P. In the third step, the outer quadrilaterals of the selected voxels are taken as a first approximation of the surface. This resembles the cuberille approach of volume visualization [HL79].

In order to get a more pleasant representation, the surface is transferred into a triangular mesh by diagonally splitting each quadrilateral into two triangles. The cuberille artifacts are smoothed using a low–pass filter that assigns a new position to each vertex of a triangle. This position is computed as the weighted average of its old position and the position of its neighbors. The approximation of the resulting surface is improved by warping it towards the data points. For more on that we refer to Section 2.3.2.

The Approach of Hoppe et al.

Another possibility of surface–oriented cell selection is based on the distance function approach of Hoppe [HDD+92, HDD+93, Hop94].

The distance function of the surface of a closed object tells for each point in space its minimum signed distance to the surface. Points on the surface of course have distance 0, whereas points outside the surface have positive, and points inside the surface have negative distance. The calculation of the distance function is outlined in Section 2.2.1.

The first step of the algorithm again is implemented by a regular voxel grid. The voxel cells selected in the second step are those which have vertices of opposite sign. Evidently, the surface has to traverse these cells. In the third step, the surface is obtained by the marching cubes algorithm of volume visualization [LC87]. The marching cubes algorithm defines templates of separating surface patches for each possible configuration of the signs of the distance values at the vertices of a voxel cell. The voxels are replaced with these triangulated patches. The resulting triangular mesh separates the positive and negative distance values on the grid.

A similar algorithm has been suggested by Roth and Wibowoo [RW97]. It differs from the approach of Hoppe et al. in the way the distance function is calculated, cf. Section 2.2.1. Furthermore, the special cases of profile lines and multiple view range data are considered besides scattered point data.

A difficulty with these approaches is the choice of the resolution of the voxel grid. One effect is that gaps may occur in the surface because of troubles of the heuristics of distance function calculation.

The Approach of Bajaj, Bernardini et al.

The approach of Bajaj, Bernardini et al. [BBX95] differs from the previous ones in that spatial decomposition is now irregular and adaptive.

The algorithm also requires a signed distance function. For this purpose, a first approximate surface is calculated in a preprocessing phase. The distance to this surface is used as distance function. The approximate surface is calculated using α–solids which will be explained in Section 2.1.2.

Having the distance function in hand, the space is incrementally decomposed into tetrahedra starting with an initial tetrahedron surrounding the whole data set. The tetrahedra traversed by the surface are found by inspecting the sign of the distance function at the vertices. For each of those tetrahedra, an approximation of the traversing surface is calculated. For this purpose, a Bernstein–Bézier trivariate implicit approximant is used. The approximation error to the given data points is calculated. A bad approximation induces a further refinement of the tetrahedrization. The refinement is performed by incrementally inserting the centers of tetrahedra with high approximation error into the tetrahedrization. The process is iterated until a sufficient approximation is achieved.

In order to keep the shape of the tetrahedra balanced, an incremental tetrahedrization algorithm is used so that the resulting tetrahedrizations are always Delaunay tetrahedrizations. A tetrahedrization is a *Delaunay tetrahedrization* if none of its vertices lies inside the circumscribed sphere of any of its tetrahedra [PS85].

The resulting surface is composed of tri-variate implicit Bernstein–Bézier patches. A C^1-smoothing of the constructed surfaces is obtained by applying a Clough–Tocher subdivision scheme.

In Bernardini et al. [BBCS97, Ber96] an extension and modification of this algorithm is presented [BBX97, BB97]. The algorithm consists of an additional mesh simplification step in order to reduce the complexity of the mesh represented by the α–solid [BS96]. The reduced mesh is used in the last step of the algorithm for polynomial–patch data fitting using Bernstein–Bézier patches for each triangle by interpolating the vertices and normals and by approximating data points in its neighborhood. Additionally, the representation of sharp features can be achieved in the resulting surface.

Edelsbrunner's and Mücke's α–shapes

Edelsbrunner and Mücke [EM94, Ede92] also use an irregular spatial decomposition. In contrast to the previous ones, the given sample points are part of the subdivision. The decomposition chosen for that purpose is the Delaunay tetrahedrization of the given set P of sample points. A tetrahedrization of a set P of spatial points is a decomposition of the convex hull of P into tetrahedra so that all vertices of the tetrahedra are points in P. It is well known that each finite point set has a Delaunay tetrahedrization which can be calculated efficiently [PS85]. This is the first step of the algorithm.

The second step is to remove tetrahedra, triangles, and edges of the Delaunay tetrahedrization using so–called α–balls as eraser sphere with radius α. Each tetrahedron, triangle, or edge of the tetrahedrization whose corresponding minimum surrounding sphere does not fit into the eraser sphere is eliminated. The resulting so–called *α–shape* is a collection of points, edges, faces, and tetrahedra.

In the third step, triangles are extracted out of the α–shape which belong to the desired surface, using the following rule. Consider the two possible spheres of radius α through all three points of a triangle of the α–shape. If at least one of these does not contain any other point of the point set, the triangle belongs to the surface.

A problem of this approach is the choice of a suitable α. Since α is a global parameter the user is not swamped with many open parameters, but the drawback is that a variable point density is not possible without loss of detail in the reconstruction. If α is too small, gaps in the surface can occur, or the surface may become fragmented.

Guo et al. [GMW97] also use α–shapes. They propose a so–called *visibility* algorithm for extracting those triangles out of the α–shape which represent the simplicial surface.

Another approach using the principle of α–shapes has been presented by Teichmann et al. [TC98]. Here, the basic α–shape algorithm is extended by *density scaling* and by *anisotropic–shaping*. Density scaling is used to vary the value of α according to the local density of points in a region of the data site. Anisotropic–shaping changes the form of the α–ball which is based on point normals. The α–balls become "ellipsoidal" that allows a better adaption to the flow of the surface. Using these principles the adaptiveness of α–shapes could be improved.

Attali's Normalized Meshes

In the approach of Attali [Att97], the Delaunay tetrahedrization is also used as a basic spatial decomposition. Attali introduces so–called normalized meshes which are contained in the Delaunay graph. It is formed by the edges, faces and tetrahedra whose dual element of the Voronoi diagram intersects the surface of the object. The *Voronoi diagram* of a point set P is a partition of the space in regions of nearest neighborhood. For each point $\mathbf{p}$ in P, it contains the region of all points in space that are closer to $\mathbf{p}$ than to any other point of P.

In two dimensions, the normalized mesh of a curve c consists of all edges between pairs of points of the given set P of sample points on c which induce an edge of the Voronoi diagram of P that intersects

c. The nice property of normalized meshes is that for a wide class of curves of bounded curvature, the so–called r–regular shapes, a bound on the sample density can be given within which the normalized mesh retains all the topological properties of the original curve.

For reconstruction of c, the edges belonging to the reconstructed mesh are obtained by considering the angle between the intersections of the two possible circles around a Delaunay edge. The angle between the circles is defined to be the smaller of the two angles between the two tangent planes at one intersection point of the two circles. This characterization is useful because Delaunay discs tend to become tangent to the boundary of the object. The reconstructed mesh consists of all edges whose associated Delaunay discs have an angle smaller than $\frac{\pi}{2}$. If the sample density is sufficiently high, the reconstructed mesh is equal to the normalized mesh.

While in two dimensions the normalized mesh is a correct reconstruction of shapes having the property of r–regularity, the immediate extension to three dimensions is not possible. The reason for that is that some Delaunay spheres can intersect the surface without being approximately tangent to it. Therefore, the normalized mesh in three dimensions does not contain all faces of the surface.

To overcome this problem, two different heuristics for filling the gaps in the surface structure have been introduced. The first heuristic is to triangulate the border of a gap in the triangular mesh by considering only triangles contained in the Delaunay tetrahedrization. The second heuristic is volume–based. It merges Delaunay tetrahedra to build up the possibly different solids represented in the point set. The set of mergeable solids is initialized with the Delaunay tetrahedra and the complement of the convex hull. The merging step is performed by processing the Delaunay triangles according to decreasing diameters. If the current triangle separates two different solids in the set of mergable solids, they are merged if the following holds:

- no triangle from the normalized mesh disappears,
- merging will not isolate sample points inside the union of these objects, i.e. the sample points have to remain on the boundary of at least one object.

The surface finally yielded by the algorithm is formed by the boundary of the resulting solids.

Weller's Approach of Stable Voronoi Edges

Let P be a finite set of points in the plane. P' is an *ε–perturbation* of P if $d(\mathbf{p}_i, \mathbf{p}'_i) \leq \varepsilon$ holds for all $\mathbf{p}_i \in P$, $\mathbf{p}'_i \in P'$, $i = 1, \ldots, n$. An edge $\overline{\mathbf{p}_i\mathbf{p}_j}$ of the Delaunay triangulation is called *stable* if the perturbed endpoints $\mathbf{p}'_i$, $\mathbf{p}'_j$ are also connected by an edge of the Delaunay triangulation of the perturbed point set P'.

It turns out that for intuitively reasonably sampled curves in the plane, the stable edges usually are the edges connecting two consecutive sample points on the curve, whereas the edges connecting non–neighboring sample points are instable. The stability of an edge can be checked in time $\mathcal{O}(\#\text{Voronoi neighbors})$, cf. [Wel97].

The extension of this approach to 3D–surfaces shows that large areas of a surface can usually be reconstructed correctly, but still not sufficiently approximated regions do exist. This resembles the experience reported by Attali [Att97], cf. Section 2.1.1. Further research is necessary in order to make stability useful for surface construction.

The Voronoi Filtering Approach of Amenta and Bern

The idea of the *Voronoi filtering* approach [ABK98, AB98] is to extract a so–called *crust* out of the set of Voronoi vertices combined with the original point set.

In two dimensions the algorithm can be described as follows. First the Delaunay triangulation of $P \cup V$ is determined where V is the set of its Voronoi vertices of the Voronoi diagram of P. From the resulting Delaunay triangulation the so–called *crust* is extracted which consists of the Delaunay edges connecting points of P.

An interesting observation is that the crust is also part of the Delaunay triangulation of the input point set P. The additional Voronoi vertices are needed to eliminate undesired edges from the Delaunay triangulation, by the property of the Delaunay edges that their circumcircles are empty of points in $P \cup V$. This process is called *Voronoi filtering*.

A sampling theorem based on the medial axis has been formulated for this algorithm. "Sampling theorem" means a characterization of sample point sets for which an algorithm yields a correct surface. The medial axis consists of all points which are centers of spheres that touch a given surface in at least two points.

A difficulty with the extension of this algorithm to three dimensions is that, while in two dimensions the Voronoi vertices of a sufficiently dense data set are located near the medial axis, this is not necessarily the case in three–dimensional space. In order to cope with this difficulty, for each sample point **p** the following calculations are performed:

- If **p** does not lie on the convex hull of P then the Voronoi vertex $\mathbf{v}^+$ of the Voronoi cell $V_\mathbf{p}$ of **p** is computed which is the farthest from **p**. The vector $\mathbf{n}^+ := \overrightarrow{\mathbf{pv}^+}$ points in the direction from **p** to $\mathbf{v}^+$.
- If **p** lies on the convex hull then $\mathbf{n}^+$ is taken as the average of the outer normals of the adjacent triangles. $\mathbf{v}^-$ is defined as the Voronoi vertex of $V_\mathbf{p}$ with negative projection on $\mathbf{n}^+$ that is farthest from **p**.

The points $\mathbf{v}^-$ and $\mathbf{v}^+$ are denoted as *poles*. The set V of the poles takes over the role of the set V of Voronoi vertices of the two–dimensional algorithm. This means that the Delaunay tetrahedrization DT of $P \cup V$ is computed, and a "*crust*" is extracted which is defined by all triangles in $\mathrm{DT}(P \cup V)$ for which all three vertices are sample points of P.

The crust usually does not describe a piecewise linear manifold. It may contain additional triangles which have to be removed in a further filtering phase. In [AB98] so–called *normal filtering* has been suggested where all triangles are eliminated which have normals deviating too much from $\mathbf{n}^+$ or $\mathbf{n}^-$. Still existing superfluous triangles are eliminated in a final post–processing step.

The Short Crust Algorithm of Amenta and Choi

In a more recent approach [AC99, ACDL00], called *short crust algorithm*, Amenta et al. replace the *normal filtering* by a simpler algorithm with just a single Voronoi diagram computation.

The algorithm starts by computing a normal at each sample point. The normal is estimated by using "poles" as in their first approach [AB98]. For each Voronoi cell $V_\mathbf{p}$, the Voronoi vertex **v** farthest from the sample point **p** is taken as a pole. The line through **p** and its pole **v** is almost normal to S and is called the *estimated normal line* at **p**. For an angle θ the *co–cone* at **p** is computed. The co–cone is the complement of the double cone with apex **p** making an angle of $\pi/2 - \theta$ with the estimated normal line at **p**. Then those Voronoi edges are determined which intersect the co–cones of all three sample points inducing the Voronoi regions incident to the edge. The dual triangles of these edges form a candidate set T.

A subsequent *manifold extraction step* derives a piecewise linear manifold from T by recursively removing any triangle in T adjacent to a *sharp edge*. A sharp edge is one for which the angle between two adjacent triangles is sharp, that is, in circular order is greater than $3\pi/2$. In practice this recursive deletion of triangles might be problematic because it can remove sucessively all triangles of T. A heuristic called *umbrella check* is used in order to prevent this problem: triangles at sharp edges are only deleted if their three vertices all have *umbrellas*. A vertex **v** is called to have an umbrella if there exists a set of triangles incident to **v** which form a topological disc and no two consecutive triangles around the disc meet at a dihedral angle less than $\frac{\pi}{2}$ or more than $\frac{3\pi}{2}$. The dihedral angle is the smaller one of the two angles between the planes of the triangles at their line of intersection.

Umbrella Filter Algorithm by Adamy, Giesen, and John

The so–called umbrella filter algorithm of Adamy et al. [AGJ00, AGJ01] starts with the Delaunay tetrahedrization of the sample point set P. Then at each point $\mathbf{p} \in P$ an "umbrella" is computed. An umbrella is a sequence of triangles incident to a point which is homeomorphic to a two–dimensional closed disc and which does not have **p** as a point of its border. After that, all triangles that do not belong to an umbrella are deleted. From the resulting set of triangles, superfluous triangles are eliminated in a topological clean–up phase, in order to get a manifold. Possibly occuring holes in the mesh are closed in a final hole–filling phase.

An umbrella is formed over special triangles called *Gabriel triangles*. The triangles are chosen with increasing value of their lower λ–interval bound until the set of chosen triangles contains an umbrella. The λ–interval boundaries λ_1 and λ_2 of a triangle t are calculated by $\lambda_i := diam(t)/diam(t_i)$, $i = 1, 2$, where t_i are the two incident tetrahedra of t in the Delaunay triangulation (if t is on the convex hull the values of the missing tetrahedron is set to 0). The interval boundaries are the minimum and the maximum of λ_1 and λ_2.

The topological clean up is performed by distinguishing between three types of triangles which hurt the umbrella condition. Each type is treated by a deletion procedure.

Holes are filled by formulating topological surface conditions and boundary constraints as linear inequalities so that the solution with integer values specifies a topologically correct surface filling the hole.

2.1.2 Volume–Oriented Cell Selection

Volume–oriented cell selection also consists of three steps which at a first glance are quite similar to those of surface–oriented selection:

Volume–oriented cell selection:

1. Decompose the space in cells.
2. Remove those cells that do not belong to the volume bounded by the sampled surface.
3. Calculate a surface from the selected cells.

The difference is that a volume representation, in contrast to a surface representation, is obtained.

Most implementations of volume–oriented cell selection are based on the Delaunay tetrahedrization of the given set P of sample points. The algorithms presented in the following differ in how volume–based selection is performed. Some algorithms eliminate tetrahedrons that are expected to be outside the desired solid, until a description of the solid is achieved [Boi84, IBS97, Vel94]. Another methodology is the use of the Voronoi diagram in order to describe the constructed solid by a "skeleton" [SB97, Att97].

Boissonnat's Volume–Oriented Approach

Boissonnat's volume–oriented approach starts with the Delaunay triangulation of the given set P of sample points. From this triangulation of the convex hull, tetrahedra having particular properties are successively removed. First of all, only tetrahedra with *two faces, five edges and four points* or *one face, three edges and three points* on the boundary of the current polyhedron are eliminated. Because of this elimination rule only objects without holes can be reconstructed, cf. Figure 2.1. Tetrahedra

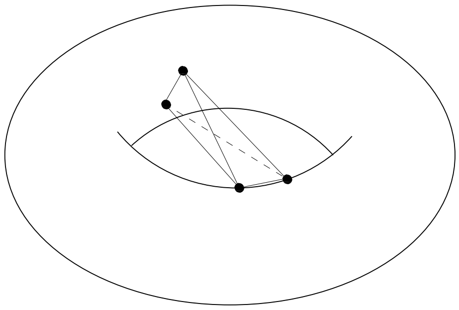


Figure 2.1: Boissonnat's volume–oriented approach. An example for a tetrahedron which cannot be removed by the elimination rule of Boissonnat. The tetrahedron in the hole of the torus has four faces on the boundary.

of this type are iteratively removed according to decreasing *decision values*. The decision value is the maximum distance of a face of the tetrahedron to its circumsphere. This decision value is useful because flat tetrahedra of the Delaunay tetrahedrization usually tend to be outside the object and cover areas of higher detail. The algorithm stops if all points lie on the surface, or if the deletion of the tetrahedron with highest decision value does not improve the sum taken over the decision values of all tetrahedra incident to the boundary of the polyhedron.

The Extended Gabriel Hypergraph Approach of Attene and Spagnuolo

The algorithm of [AS00] starts with the generation of the Delaunay tetrahedrization $DT(P)$ of the given point set P. Then, similar to Boissonnat's approach [Boi84], tetrahedra are iteratively removed from the polyhedron until all vertices lie on the boundary of the polyhedron. This process is called sculpturing.

Sculpturing can either be *constrained* or *non–constrained*. For non–constrained sculpturing a tetrahedron t is removable if it fulfills the criteria of removal of Boissonat's approach [Boi84].

Constraint sculpturing uses so–called *extended Gabriel hypergraphs (EGH)*. An EGH is constructively derived from the Gabriel graph $GG(P)$. The Gabriel graph (GG) consists of all edges $\overline{\mathbf{pq}}$ between points $\mathbf{p}, \mathbf{q}$ of P for which the smallest diameter sphere does not contain any other point of P. Initially $EGH(P) = (P, E_{EGH}, T)$ where $T := \emptyset$. Then, E_{EGH} is successively extended by edges $\overline{\mathbf{qr}}$ for which incident edges $e_1 = \overline{\mathbf{pq}}, e_2 = \overline{\mathbf{pr}}$ in E_{EGH} exist which are not collinear and for which the smallest diameter ball around $\mathbf{p}, \mathbf{q}, \mathbf{r}$ does not contain any other point of P. This process is iterated until no further edge can be added to E_{EGH}. Any cycle of three edges of E_{EGH} defines a triangle of T.

For constraint sculpturing, a tetrahedron t is removable if the following two rules are satisfied:

- if t has just one face f on the boundary then f must not belong to $EGH(P)$.

- if t has two faces f_1, f_2 on the boundary then f_1, f_2 must not belong to the EGH. Additionally, the common edge e of f_1, f_2 must not belong to $EMST(P)$.

The sculpturing process starts with constraint sculpturing, and tetrahedra having the longest edge on the boundary are removed first. The reason is that the connection between very distant vertices is less probable than that of two very close points. If the goal that all vertices are on the boundary is not achieved by constraint sculpturing, what may be the case for badly sampled points, constrained sculpturing is followed by non–constrained sculpturing.

In order to recover holes, a process of *non–constrained sculpturing with EMST restriction* follows. This happens if not all edges of the EMST are on the boundary. The process is similar to *non–constrained sculpturing* but is restricted to all removable tetrahedra whose removal adds an EMST edge to the boundary. Afterwards, a *hole recovering process* is applied. Its task is to remove so–called *pseudo–prisms*. A pseudo–prism is a set of three adjacent tetrahedra which remain in the region of a hole because each one of them cannot be classified as removable with the above criterions.

The Approach of Isselhard, Brunnett, and Schreiber

The approach of [IBS97] is an improvement of the volume–oriented algorithm of Boissonnat [Boi84]. While Boissonnat cannot handle objects with holes, the deletion procedure of this approach is modified so that construction of holes becomes possible.

As before, the algorithm starts with the Delaunay triangulation of the point set. An incremental tetrahedron removal procedure is then performed on tetrahedra at the boundary of the polyhedron, as in Boissonnat's algorithm. The difference is that more types of tetrahedra can be removed, namely those with *one face and four vertices*, or *three faces*, or *all four faces* on the current surface provided that no point would become isolated by elimination.

The elimination process is controlled by an *elimination function*. The elimination function is defined as the maximum decision value (in the sense of Boissonnat) of the remaining removable tetrahedra. In this function, several significant jumps can be noticed. One of these jumps is expected to indicate that the desired shape is reached. In practice, the jump before the stabilization of the function on a higher level is the one which is taken. This stopping point helps handling different point densities in the sample set which would lead to undesired holes caused by the extended set of removable tetrahedra in comparison to Boissonnat's algorithm [Boi84].

If all data points are already on the surface, the algorithm stops. If not, more tetrahedra are eliminated in order to recover sharp edges (reflex edges) of the object. For that purpose the elimination rules are restricted to those of Boissonnat, assuming that all holes present in the data set have been recovered at this stage. Additionally, the decision value of the tetrahedra is scaled by the radius of the circumscribed sphere as a measure for the size of the tetrahedron. In this way, the cost of small tetrahedra is increased which are more likely to be in regions of reflex edges than large ones. The elimination continues until all data points are on the surface and the elimination function does not decrease anymore.

The γ–indicator Approach of Veltkamp

In order to describe the method of Veltkamp [Vel94, Vel95] some terminology is required. A *γ–indicator* is a value associated to a sphere through three boundary points of a polyhedron which is positive or negative. Its absolute value is computed as $1 - \frac{r}{R}$, where r is the radius of the surrounding circle of the boundary triangle and R the radius of the surrounding sphere of the boundary tetrahedron. For γ–indicator the value is taken negative if the center of the sphere is on the inner side, and positive if the center is on the outer side of the polyhedron. Note, that the γ–indicator is independent of the size of the boundary triangle (tetrahedron, respectively). Therefore, it adapts to areas of changing point density. A *removable face* is a face with positive γ–indicator value.

The first step of the algorithm is to calculate the Delaunay tetrahedrization. In the second step, a heap is initialized with removable tetrahedra which are sorted according to their γ–indicator value. The removable tetrahedra are of the same boundary types as in Boissonnat's volume–oriented approach [Boi84]. The tetrahedron with the largest γ–indicator value is removed and the boundary is updated. This process continues until all points lie on the boundary, or no further removable tetrahedra exist.

The main advantage of this algorithm is the adaptation of the γ–indicator value to variable point density. Like Boissonnat's approach, the algorithm is restricted to objects without holes.

The Approach of Schreiber and Brunnett

The approach of Schreiber and Brunnett [Sch97, SB97] uses properties of the Voronoi diagram of the given sample point set for tetrahedra removal. One property is that the Voronoi diagram is dual to the Delaunay tetrahedrization of a given point set. Each vertex of the Voronoi diagram corresponds to the center of a tetrahedron of the tetrahedrization. Edges of the Voronoi diagram correspond to neighboring faces of the tetrahedra dual to its vertices. The same observation holds for Voronoi diagrams in the plane which are used in the following explanation of the 2D–version of the algorithm.

In the first step, the Delaunay triangulation and the Voronoi diagram of P are determined. The second step, selection of tetrahedra, uses a minimum spanning tree of the Voronoi graph. The *Voronoi graph* is the graph induced by the vertices and edges of the Voronoi diagram. A *minimum spanning tree* (MST) of a graph is a subtree of the graph which connects all vertices and has minimum summed edge length. Edge length in our case is the Euclidean distance of the two vertices of the edge. A pruning strategy is applied which possibly decomposes the tree into several disjoint subtrees. Each subtree represents a region defined by the union of the triangles dual to its vertices.

Two pruning rules have been developed for that purpose:

1. All those edges are removed for which no end point is contained in the circumcircle of the dual Delaunay triangle of the other end point.
2. An edge is removed if its length is shorter than the mean value of the radii of both circumcircles of the dual Delaunay triangles of its Voronoi end points.

The number of edges to be eliminated is controlled by using the edge length as a parameter.

The resulting regions are then distinguished into inside and outside. In order to find the inside regions, we add the complement of the convex hull as further region to the set of subtree regions. The algorithm starts with a point on the convex hull which is incident to exactly two regions. The region different from the complement of the convex hull is classified "inside". Then the label "inside" is propagated to neighboring regions by again considering points that are incident to exactly two regions. After all regions have been classified correctly, the boundary of the constructed shape is obtained as the boundary of the union of the region labeled "inside". An adaptation of this method to three dimensions is possible.

The α–solids of Bajaj, Bernardini et al.

Bajaj, Bernardini et al. [BBX95, BBX97, BB97, BBCS97] calculate so–called *α–solids*. While α–shapes are computed by using eraser spheres at every point in space, the eraser spheres are now applied from outside the convex hull, like in Boissonnat's approach [Boi84]. In order to overcome the approximation problems inherent to α–shapes a re–sculpturing scheme has been developed. Re–sculpturing roughly follows the volumetric approach of Boissonnat in that further tetrahedra are removed. The goal is to generate refined structures of the object provided the α–shape approach has correctly recognized the coarse structures of the shape.

2.2 Surface Construction with Distance Functions

The distance function of a surface gives the shortest distance of any point in space to the surface. For closed surface the distances can be negative or positive, depending on whether a point lies inside or outside of the volume bounded by the surface. In the preceding section, we have already described an algorithm which uses the distance function for the purpose of surface construction, but the question of distance function calculation has been left open. Solutions are presented in the next subsection.

Besides marching cubes construction of surfaces as explained in Section 2.1.1, distance plays a major role in construction of surfaces using the medial axis of a volume. The medial axis of a volume consists of all points inside the volume for which the maximal sphere inside the volume and centered at this point does not contain the maximal sphere of any other point. Having the medial axis and the radius of the maximum sphere at each of its points, the given object can be represented by the union taken over all spheres centered at the skeleton points with the respective radius. An algorithm for surface construction based on medial axes is outlined in Section 2.2.1.

A further application of the distance function [BBX95] is to improve the quality of a reconstructed surface.

2.2.1 Calculation of Distance Functions

The Approach of Hoppe et al.

Hoppe et al. [HDD$^+$92, Hop94] suggest the following approach. At the beginning, for each point $\mathbf{p}_i$ an estimated tangent plane is computed. The tangent plane is obtained by fitting the best approximating plane in the least square sense [DH73] into a certain number k of points in the neighborhood of $\mathbf{p}_i$. In order to get the sign of the distance in the case of closed surfaces, a consistent orientation of neighboring tangent planes is determined by computing the *Riemannian graph*. The vertices of the Riemannian graph are the centers of the tangent planes which are defined as the centroids of the k points used to calculate the tangent plane. Two tangent plane centers $\mathbf{o}_i, \mathbf{o}_j$ are connected with an edge $\overline{\mathbf{o}_i\mathbf{o}_j}$ if one center is in the k–neighborhood of the other center. By this construction, the edges of the Riemannian graph can be expected to lie close to the sampled surface. Each edge is weighted by 1 minus the absolute value of the scalar product between normals of the two tangent plane centers defining the edge. The orientation of the tangent planes is determined by propagating the orientation at a starting point by traversing the minimum spanning tree of the resulting weighted Riemannian graph.

Using the tangent plane description of the surface and their correct orientations, the signed distance is computed by first determining the tangent plane center nearest to the query point. Its amount is given by the distance between the query point and its projection on the nearest tangent plane. The sign is obtained from the orientation of the tangent plane.

The Approach of Roth and Wibowoo

The goal of the algorithm of Roth and Wibowoo [RW97] is to calculate distance values at the vertices of a given voxel grid surrounding the data points. The data points are assigned to the voxel cells into which they fall. An ”outer” normal vector is calculated for each data point by finding the closest two neighboring points in the voxel grid, and then using these points along with the original point to compute the normal.

The normal orientation which is required for signed distance calculation is determined as follows. Consider the voxel grid and the six axis directions $(\pm x, \pm y, \pm z)$. If we look from infinity down each axis into the voxel grid, then those voxels that are visible must have their normals point towards the viewing direction. The normal direction is fixed for these visible points. Then the normal direction is propagated to those neighboring voxels whose normals are not fixed by this procedure. This heuristic only works if the non-empty voxels define a closed boundary without holes.

The value of the signed distance function at a vertex of the voxel grid is computed as the weighted average of the signed distances of every point in the eight neighboring voxels. The signed distance of a point with normal is the Euclidean distance to this point, with positive sign if the angle between the normal and the vector towards the voxel vertex exceeds $90°$.

Bittar's et al. Surface Construction by Medial Axes

The approach of Bittar et al. [BTG95] consists of two steps, the calculation of the medial axis and the calculation of an implicit surface from the medial axis.

The medial axis is calculated from a voxelization of a bounding box of the given set of points. The voxels containing points of the given point set P are assumed to be boundary voxels of the solid to be constructed. Starting at the boundary of the bounding box, voxels are successively eliminated until all boundary voxels are on the surface of the remaining voxel volume. A distance function is propagated from the boundary voxels to the inner voxels of the volume, starting with distance 0 on the boundary voxels. The voxels with locally maximal distance value are added to the medial axis.

The desired surface is calculated by distributing centers of spheres on the medial axis. The radius of a sphere is equal to the distance assigned to its center on the medial axis. For each sphere, a field function is defined which allows to calculate a scalar field value for arbitrary points in space. A field function of the whole set of spheres is obtained as sum of the field functions of all spheres. The implicit surface is defined as an iso–surface of the field function, that is, it consists of all points in space for which the field function has a given constant value.

In order to save computation time, a search strategy is applied which restricts the calculation of the sum to points with suitable positions.

The shape of the resulting surface is strongly influenced by the type of field function. For example, a *sharp* field function preserves details while a *soft* function smoothes out the details. Also the connectness of the resulting solid can be influenced by the shape function.

Because of the voxelization, a crucial point is tuning the resolution of the medial axis. If the resolution of the axis is low, finer details are not represented very accurately. If it is high, the detail construction is improved, but the surface may fall into pieces if the resolution is higher than the sample density.

The Power Crust Algorithm of Amenta and Choi

The latest reconstruction algorithm of Amenta al. [ACK01], the *power crust algorithm*, uses an approximation of the medial axis transformation of volumes. The approximation is defined by the union of *polar balls* which are a subset of the Voronoi balls of the sample point set P. The *poles* $\mathbf{o}_1, \mathbf{o}_2$ of a sample point $\mathbf{p}$ are the two vertices of its Voronoi cell farthest from $\mathbf{p}$, one on either side of the surface. The corresponding polar balls are the Voronoi balls $B_{\mathbf{o}_1,\rho_1}, B_{\mathbf{o}_2,\rho_2}$ with $\rho_i = d(\mathbf{o}_i, \mathbf{p})$.

The polar balls belong to two sets from which one is more or less filling up the inside of the object, and the other the outside.

The main part of the algorithm is to divide the set of polar balls into a set of inner balls which is filling up the inside of the object, and a set of outer polar balls which are outside the surface. A weighted Voronoi diagram, the *power diagram*, for the polar balls is used for that purpose. The power diagram divides the space into polyhedral cells, each cell consisting of the points in $\mathbb{R}^3$ closest to a particular ball, under a special distance function, called the *power distance*. The power diagram induces an adjacency relation between polar balls in that two balls are adjacent which have adjacent power diagram cells. The inside and outside sets of balls are obtained by a labeling procedure which uses this adjacency.

Finally, the piecewise–linear surface separating the cells of the power diagram belonging to inner polar balls from the cells belonging to outer polar balls is determined. This so–called *power crust* is the result of the algorithm.

2.3 Surface Construction by Warping

Warping–based surface construction means to deform an initial surface so that it gives a good approximation of the given point set P. For example, let the initial shape be a triangular surface. To some or all of its vertices, corresponding points in P are determined to which the vertices have to be moved in the warping process. When moving the vertices of the mesh to their new locations, the rest of the mesh is also deformed and yields a surface approximation of the points in P.

Surface construction by warping is particularly suited if a rough approximation of the desired shape is already known. This simplifies detection of corresponding points.

Several methods of describing deformable surfaces have been developed in the past. Muraki suggested a "*blobby model*" in order to approximate 2.5–D range images [Mur91]. Terzopoulos, Witkin and Kass [TM91, TWK88] made use of *deformable superquadrics* which have to fit the input data points. Miller et al. [MBL$^+$91] extract a topologically closed polyhedral model from a volume data set. The algorithm starts with a simple polyhedron that is already topologically closed. The polyhedron is deformed by growing or shrinking it so that it adapts to the object in the volume without changing its topology, according to a set of constraints. A function is associated with every vertex of the polyhedron, which associates costs with local deformation adherent to properties of simple polyhedra, and the relationship between noise and feature. By minimizing these constraints, an effect similar to inflating a balloon within a container or collapsing a piece of shrink wrap around the object is achieved.

A completely different approach to warping is modeling with *oriented particles*, suggested by Szeliski and Tonnesen [ST92]. Each particle owns several parameters which are updated during the modeling simulation. By modeling the interaction between the particles themselves the surface is being modeled using forces and repulsion. As an extension Szeliski and Tonnesen describe how their algorithm can be extended for automatic 3D reconstruction. At each sample location one particle with appropriate parameters is generated. The gaps between the sample points (particles, respectively) are filled by growing particles away from isolated points and edges. After having a rough approximation of the current surface the other particles are rejusted to smooth the surface.

In the following three subsections three approaches are outlined which stand for basically different methodologies: a purely geometric approach, a physical approach, and a computational intelligence approach.

2.3.1 Spatial Free Form Warping

The idea of spatial free–form warping is to deform the whole space in which an object to be warped is embedded in, with the effect that the object is warped at the same time. Space deformation is defined by a finite set of displacement vectors consisting of pairs of initial and target point, from which a spatial displacement vector field is interpolated using a scattered data interpolation method. A considerable number of scattered data interpolation methods is known in literature, cf. e.g. [HL93], from which those are chosen which yield the most reasonable shape for the particular field of application.

The resulting displacement vector field tells for each point in space its target point. In particular, if the displacement vector field is applied to all vertices of the initial mesh, or of a possibly refined one, the mesh is warped towards the given data points [RM95].

The advantage of spatial free form warping is that usually only a small number of control displacement vectors located at points with particular features like corners or edges is necessary. A still open question is how to find good control displacement vectors automatically.

2.3.2 The Approach of Algorri and Schmitt

The idea of Algorri and Schmitt [AS96] is to translate a given approximated triangular mesh into a physical model. The vertices of the mesh are interpreted as mass points. The edges represent springs. Each nodal mass of the resulting mesh of springs is attached to its closest point in the given set P of sample points by a further spring. The masses and springs are chosen so that the triangular mesh is deformed towards the data points.

The model can be expressed as a linear differential equation of degree 2. This equation is solved iteratively. Efficiency is gained by embedding the data points and the approximate triangular mesh into a regular grid of voxels, like that one already yielded by the surface construction algorithm of the same authors, cf. Section 2.1.1.

2.3.3 Kohonen Feature Map Approach of Baader and Hirzinger

The Kohonen feature map approach of Baader and Hirzinger [BH93, BH94, Baa95] can be seen as another implementation of the idea of surface construction by warping. Kohonen's feature map is a two–dimensional array of units (neurons). Each unit u_j has a corresponding weight vector $\mathbf{w}_j$. In the beginning these vectors are randomly chosen with length equal to 1.

During the reconstruction or training process the neurons are fed with the input data which affect their weight vectors (which resemble their position in 3D space). Each input vector $\mathbf{i}$ is presented to the units u_j which produce an output o_j of the form

$$o_j = \mathbf{w}_j * \mathbf{i},$$

which is the scalar vector product of $\mathbf{w}_j$ and $\mathbf{i}$. The unit generating the highest response o_j is the center of the excitation area. The weights of this unit and a defined neighborhood are updated by

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \epsilon_j \cdot (\mathbf{i} - \mathbf{w}_j(t)).$$

After updating the weight vectors are normalized again. The value $\epsilon_j := \eta \cdot h_j$ contains two values, the learning rate η and the neighborhood relationship h_j. Units far away from the center of excitation are only slightly changed.

The algorithm has one additional difficulty. If the input point data do not properly correspond to the neuron network it is possible that some neurons might not be moved sufficiently towards the desired surface. Candidates are neurons which have not been in any center of excitation so far, and therefore have been updated just by neighborhood update which usually is not sufficient to place units near the real surface. Having this in mind, Baader and Hirzinger have introduced a kind of *reverse training*. Unlike the *normal training* where for each input point a corresponding neural unit is determined and updated, the procedure in the intermediate *reverse training* is reciprocal. For each unit u_j the part of the input data with the highest influence is determined and used for updating u_j.

The combination of normal and reverse training completes the training algorithm of Baader and Hirzinger.

2.4 Incremental Surface–Oriented Construction

The idea of incremental surface–oriented construction is to build up the interpolating or approximating surface directly on surface–oriented properties of the given data points.

For example, surface construction may start with an initial surface edge at some location of the given point set P, connecting two of its points which are expected to be neighboring on the surface. The edge is successively extended to a larger surface by iteratively attaching further triangles at boundary edges of the emerging surface. The surface–oriented algorithms of Boissonnat [Boi84] and of Gopi [GKS00] sketched in the following work according to this scheme. As the algorithm of Gopi, the *ball–pivoting algorithm* of Bernardini et. al. [BJMT99] follows the advancing–front paradigm, but it assumes that normals are given at the sampling points.

Another possibility is to calculate an initial global wire frame of the surface which is augmented iteratively to a complete surface. This is the idea of the approach presented in this thesis, and earlier versions published in [Men95, MM98a, MM98b].

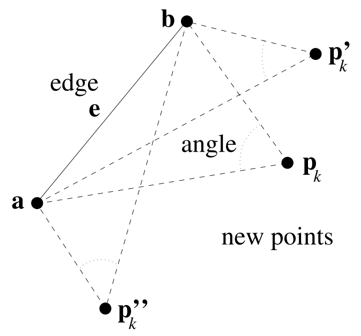


Figure 2.2: Point $\mathbf{p}_k$ sees the boundary edge **e** under the largest angle. The points are projected onto the local tangent plane of points in the neighborhood of **e**.

Boissonat's Surface–Oriented Approach

Boissonnat's surface oriented contouring algorithm [Boi84] usually starts at the shortest connection between two points of the given point set P. In order to attach a new triangle at this edge, and later on to other edges of the boundary, a locally estimated tangent plane is computed based on the points in the neighborhood of the boundary edge. The points in the neighborhood of the boundary edge are then projected onto the tangent plane. The new triangle is obtained by connecting one of these points to the boundary edge. That point is taken which maximizes the angle at its edges in the new triangle, that is, the point sees the boundary edge under the maximum angle, cf. Figure 2.2. The algorithm terminates if there is no free edge available any more. The behavior of this algorithm can be seen in Figure 2.3.

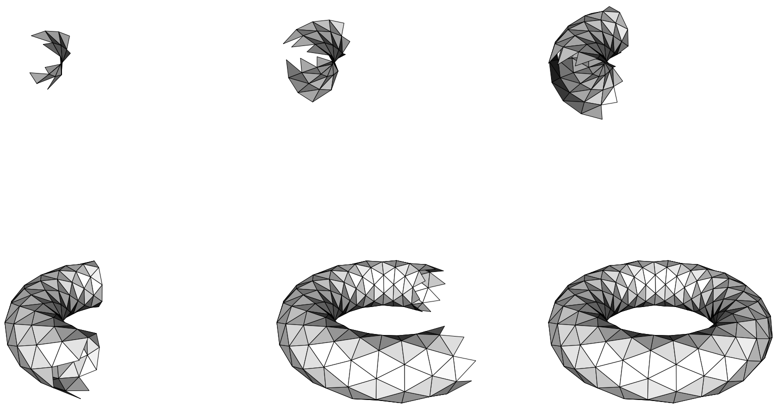


Figure 2.3: This figure shows the behavior of a contouring algorithm like Boissonnat's [Boi84] during the reconstruction of a torus. The picture sequence was not reconstructed by the original software (which was not available).

Reconstruction with Lower Dimensional Localized Delaunay Triangulation

In [GKS00] an approach using lower dimensional localized Delaunay triangulation is used for surface construction. It consists of mainly four steps: normal computation, candidate points selection, Delaunay neighbor computation and triangulation.

The normal of each sample point is computed by a simple k-nearest-neighbor approach. The normals of neighboring points are oriented consistently, so that an orientable manifold can be represented. Candidate point selection generates a set of points $P_{\mathbf{p}}$ which might be connected to a point $\mathbf{p}$ in the final triangulation, by a conglomerate of estimation functions. Delaunay neighbor computation is performed in the projection of $P_{\mathbf{p}}$ onto an estimated tangent plane $H_{\mathbf{p}}$ of $\mathbf{p}$.

In the final step, an advancing front algorithm is applied. The process starts with an initial point and all triangles that surround it are taken as initial mesh. In general, boundary points of the current triangulations and the Delaunay neighborhood information are used to extend the mesh until it is complete.

2.5 Clustering

It may happen that more than one connected shape is represented in a sample data set. In that case, most of the methods described up to now may have troubles. The difficulty can be overcome by *segmenting* or *clustering* the sample point set P into subsets of points which are likely to belong to the same component. The following approach of Fua and Sander [FS91, FS92a, FS92b] is an example of how clustering can be performed.

The Approach of Fua and Sander

The approach of Fua and Sander [FS91, FS92a, FS92b] consists of three steps. In the first step, a quadric surface patch is iteratively fitted to a local environment of every data point, and then the data point is moved onto the surface patch. An additional effect of this step besides yielding a set of local surfaces is smoothing of the given sample data.

In the second step, the sample points together with their local surface patches are moved onto positions on a regular grid.

In the third step, a surface–oriented clustering is performed. A graph is calculated which has the corrected sample points of the previous step as vertices. An edge is introduced between two vertices if the quadrics assigned to them are similar. A measure of similarity and a threshold are defined for that purpose. The connected components of the graph define the clusters of the surface in the data set.

Each of these clusters can now be treated by one of the reconstruction algorithms of the previous sections.

2.6 Discussion and Categorization of the New Approach

The strength of volume-oriented cell selection is the topological feasibility of the constructed volume. A disadvantage of the approach is that it is less suited for surface pieces not bounding a volume. Surface-oriented cell selection may be sensitive to grid resolution (voxel grid, MC surface extraction), or may cause difficulties by filtering out the right triangles from a superset of triangles obtained in a first phase, in order to achieve manifold surfaces.

Distance function approaches can be seen as a special case of the volume-oriented approach. They have similar properties.

Surface warping approaches are reliable with respect to surface topology, but the definition of the warping function is still a problem.

Incremental surface-oriented construction is suitable for surfaces not bounding a volume, which also may have boundaries and holes. Its difficulty lies in the decision which points have to be connected. Often this task is performed locally according to the advancing front scheme which may cause troubles for not very dense sample sets.

A difficulty with all solutions is that almost no characterization of point sets has been given up to now for which an algorithm is successful. Recent exceptions are the work of Amenta et al. [ABK98, AB98, AC99, ACDL00, ACK01] and Adamy et al. [AGJ01]. Successful reconstruction can be characterized by the quality of approximation of the original surface by the reconstructed surface. The quality of approximation has the aspect of correct topology and the aspect of geometric proximity. For the case of curves which has been mainly treated up to now, the derivation of ”sampling theorems” is possible in a quite straightforward way. For surfaces the problem is more severe. Recently the nearest-neighbor image has been discovered as a useful concept for describing surface approximation. This concept is also used in this thesis. For the characterization of the necessary density of sample points, the concept of the medial axis of a surface has shown to be useful.

The algorithm presented in the thesis can be categorized as surface-oriented. One of its particular advantages is that arbitrary manifold surfaces with boundaries can be constructed. They need not to be the surface of a volume. The algorithm constructs the surface incrementally, controlled by a surface-skeleton which is determined in a first step. The skeleton reduces the above-mentioned difficulty of surface-oriented approaches to decide which points have to be connected. Furthermore, in contrast to most other approaches, the new algorithm is always on the ”save side”, that is it does not have to remove superfluous surface elements.

We demonstrate the existence of sample sets for the case of surfaces of bounded curvature without boundaries, for which a reliable behavior of the algorithm can be expected. The extension to surfaces with boundaries seems possible but is not done here. Additionally, we give intuitive explanations for the good behavior of the algorithm which can be noticed also at locations of infinite curvature, that is sharp edges.

Part II

The Surface Reconstruction Algorithm

Chapter 3

Outline of the Algorithm

The algorithm presented in this thesis has the following interface:

Input: A finite set $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ of points in 3D space.

Output: A straight-line manifold 2D cell complex which has the points of P as vertices.

A manifold 2D cell complex and related concepts are defined as follows.

Definition 3.1 (2D cell complex) *A **2D cell complex (CC)** (V, E, F) is given by a set V of vertices, a set $E \subseteq V \times V$ of edges, and a set $F \subseteq V \times V \times V$ of triangles. The elements of V, E, and F are also called **cells**.*

*(1) A CC is **manifold (MCC)** if at most two triangles share a common edge.*

*(2) A CC is **connected** if for every pair of points* **p**,**q** *there is a sequence of points starting with* **p** *and ending with* **q** *in which each pair of consecutive vertices define an edge of the CC.*

*(3) A CC is a **graph** if $F = \emptyset$.*

*(4) A CC is a **mesh** if every vertex and every edge has an incident triangle.*

*(5) A CC is **not self-intersecting** if the intersection of any two cells is either empty or again a cell of the CC.*

*(6) A CC is called **geometric** if the vertices are points, the edges are curves, and the triangles are surfaces in d-dimensional space for some $d > 1$. It is **straight-line** if the curves are line segments.*

An assumption of the algorithm is that the given point set P consists of points belonging to a surface S in 3D space. The suspected surface needs not necessarily to be connected, and thus the resulting surface has to be neither. The goal is that the application of the algorithm yields a straight-line mesh which is not self-intersecting if P is a "reasonable sample" of a non-self-penetrating surface.

The algorithm consists of two main phases. In the first phase, a so-called *surface description graph* (SDG) is calculated.

A type of graph which can in principle be used as SDG is the Euclidean Minimum Spanning Tree (EMST) of the given point set P. The EMST is a (geometric) tree with P as vertices so that the sum of the Euclidean lengths of its edges is minimum over all trees connecting P. The example in Figure 3.1 shows that the edges of the EMST follow the expected surface quite reasonably. More on EMSTs is presented in Chapter 4.

Other favorable types of graphs are the β-environment graphs (β-EGs). Figure 3.2 shows an example. The advantage of β-EGs is that they are more dense than the EMST, but still have the property of a

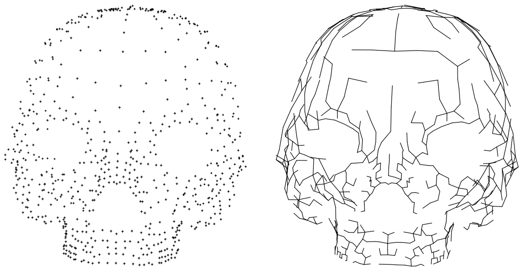


Figure 3.1: The point set and the EMST of a skull data set.

favorable surface approximation for reasonable sets of sampling points. The density can be controlled by the value of β. The β-EGs and their properties are described in Chapter 5.

Unfortunately, the EMST and the β-EG may contain edges unfavorable for reconstruction. For that reason, they need to be slightly modified by eliminating so-called *bridge edges*.

Figure 3.3 shows bridge edges connecting the two tori, but bridge edges can also be noticed for the β-EG of Figure 3.2 if the non-clustered β-environment graph is compared with its clustered variant.

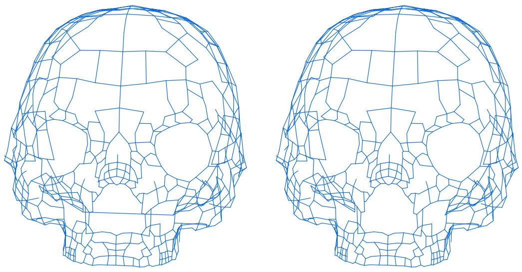


Figure 3.2: The β-environment graph of a skull for $\beta = 1$ (left) and its corresponding clustered variant (right).

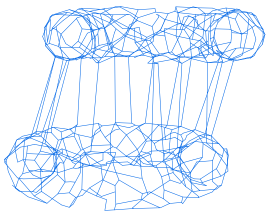


Figure 3.3: The β-environment graph of two randomly sampled tori for $\beta = 1$ without the clustering approach.

The modification applied in order to avoid bridge edges is described in Chapter 5.

In the second phase, the SDG of the first phase is embedded in space and successively augmented by edges and triangles in order to get a cell complex defining the desired reconstruction. Embedding means arranging the edges incident to every vertex into an ordered cycle. The embedding is partial in the sense that the cycle is only determined up to orientation. The sectors induced by two consecutive edges of the ordered cycles are checked for whether they can be closed to form a triangle. In many cases this is indeed possible, and the resulting triangle is reported as part of the desired manifold. However, sometimes a triangle cannot be constructed in this manner, and an alternative procedure is applied in order to complete the manifold. Figure 3.4 shows snapshots of the triangulation of the skull data set. The triangulation algorithm is described in detail in Chapter 8.

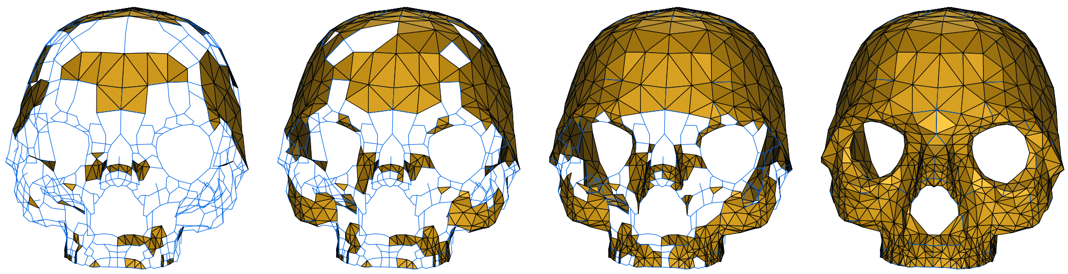


Figure 3.4: The reconstruction for the skull data set: intermediate stages of the reconstruction and the final result.

The overall approach of the algorithm follows the general principle of augmenting an initial SDG by edges and triangles so that at any time the resulting cell complex is a reasonable fitting into the data point set P. We have previously presented other solutions based on the same principle [Men95, MM98a]. There, a longer sequence of graphs has been constructed before the algorithm has switched to triangles. The advantage of the algorithm presented here is that the sequence of graphs has been reduced to one graph, due to a quite general concept of environment graphs.

The algorithm is complemented by an analysis of its reconstruction behavior. For this purpose we first give a precise definition of "reconstruction". The definition is based on the concept of nearest-neighbor (NN) image. It roughly tells that a mesh M is a reconstruction of a given surface S if the NN-image of M on S is non-self-intersecting. Then, conditions on the sample point sets are derived which are favorable in order that the mesh constructed by our algorithm is a reconstruction. These conditions are used to demonstrate that a given sample set can be augmented to a sample set for which our algorithm yields a reconstruction for closed surfaces without boundary of limited curvature, with high heuristic probability.

The philosophy of NN-embedding is subject of Chapter 6. The analysis is described in Chapter 7 for the SDG-phase, and in Chapter 9 for the phase of triangulation. Additional heuristic arguments for the favorable behavior of the algorithm in interactive modeling environments are presented in Chapter 10. Chapter 11 shows that the neighborhood information of the SDG can be used to smooth noisy data sets.

Chapter 4

The Euclidean Minimum Spanning Tree

In this chapter a number of properties of the EMST are compiled which underline the observation that the *Euclidean minimum spanning tree* (EMST) is a useful skeleton for surface-oriented interpolation of surfaces from a set of scattered points, as illustrated in Figure 4.1. Furthermore, algorithmic considerations concerning the calculation of the EMST are presented.

4.1 The Euclidean Minimum Spanning Tree – Definition and Properties

In the following we assume that the reader is familiar with the basic terminology of graph theory, like it is e.g. described in [PS85, Ede87]. Briefly, a graph $G = (V, E)$ consists of a set V of vertices and a set E of edges. The edges are defined as pairs of vertices. A path in a graph G is a sequence of different vertices so that any two consecutive vertices are connected by an edge. A graph is connected if there is a path between any two of its vertices. A cycle is a closed path, that is, its first and last vertices coincide. A tree is an acyclic connected graph, that is, a graph without cycles.

The Euclidean minimum spanning tree is defined as follows.

Definition 4.1 (Euclidean minimum spanning tree) *Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be a finite set of points in d-dimensional space. The **Euclidean minimum spanning tree** (EMST) of P is a tree that connects all points of P with edges so that the sum of its Euclidean edge lengths is minimum.*
The set of all minimum spanning trees of P is denoted by $\mathcal{EMST}(P)$.

Theorem 4.2 (Uniqueness of the EMST) *If the Euclidean distances between the points of a given finite point set P are pairwise distinct, then the EMST is unique, that is, $|\mathcal{EMST}(P)| = 1$.*

Proof: For simplicity, let in the following the union of a graph $H = (P, E)$ with an edge e, $H \cup e$, be defined as $(P, E \cup \{e\})$. The difference $H - e$ is defined analogously.

Now, let $H = (P, E)$ and $K = (P, E')$ be two different EMSTs of P. The edges in H and K are sorted in order increasing edge length.

Let e_H be the first of these edges that is in one EMST and not in the other. W.l.o.g. let e_H be in H but not in K. This means, that every edge with edge length less than e_H is in both or neither of the trees H, K.

Consider the tree $K \cup e_H$. Adding an edge to a tree always creates a cycle, so that $K \cup e_H$ must contain at least one cycle. By removing any edge of this cycle we again get a tree. By definition both trees H, K do not contain any cycles. This means that there must be at least one edge e_K in the cycle of $K \cup e_H$ which was in K but not in H. We remove that edge e_K from $K \cup e_H$ so that we get a new tree $L := K \cup e_H - e_K$.

Because e_H was the shortest edge in one tree but not in the other, and because all distances between the points are pairwise distinct, e_K is longer than e_H and cannot be of equal length.

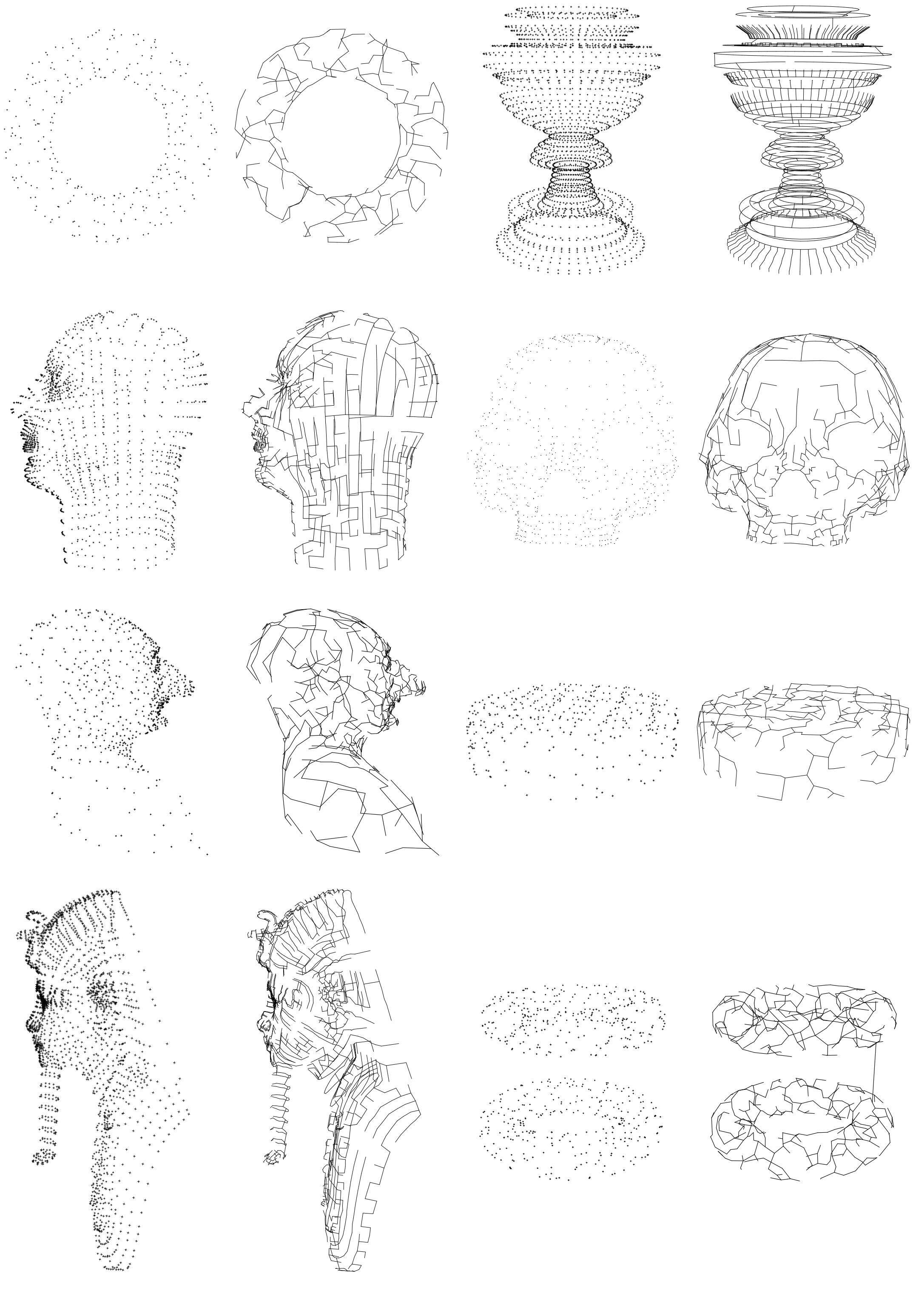
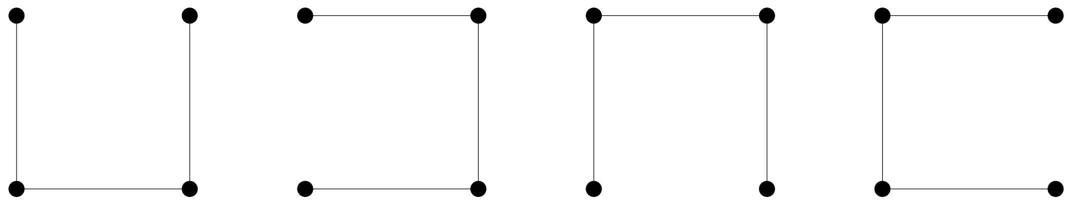


Figure 4.1: EMSTs of various point sets. The approximation of the surface is quite natural.

This means that L has an edge length sum lower than K because a longer edge e_K has been replaced by a shorter edge e_H. But because K is already an EMST, L cannot exist. Thus, H and K must be equal. ■

A case for which the EMST is not unique is displayed in Figure 4.2.



(a) (b) (c) (d)

Figure 4.2: An example of four possible EMSTs for a set of points arranged in a square.

Theorem 4.3 (EMST property of subtrees) *Let $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be a set of points in d-dimensional space and $G = (P, E) \in \mathcal{EMST}(P)$. Let $G' = (P', E')$ be the graph with a connected subset of edges $E' \subseteq E$ and $P' \subseteq P$ the set of points that is connected by edges of E'. Then, $G = (P', E') \in \mathcal{EMST}(P')$.*

Proof: Let $G' = (P', E')$ be a connected subgraph of $G = (P, E)$ with $P' \subseteq P$ and $E' \subseteq E$. If $G' = (P', E')$ is not an EMST then there exists a possibility to connect the points of P' with edges of a set E'' with shorter edge length sum. Then, because of $P' \subseteq P$ and $E' \subseteq E$ there is also a possibility to connect the points P with edges of a set $E^{new} := (E - E') \cup E''$ so that the sum of its edge lengths is lower than in E. Consequently, $G = (P, E)$ is not an EMST which is a contradiction. ■

In the following, let $d(\mathbf{p}_i, \mathbf{p}_j)$ denote the Euclidean distance between two points $\mathbf{p}_i, \mathbf{p}_j$ in d-dimensional space.
One important property of the EMST is that it connects neighboring points as stated in the next theorem.

Theorem 4.4 (Nearest neighbor property of the EMST) *Let $\mathbf{p}_i$ be an arbitrary point of a finite point set P in d-dimensional space which has a unique nearest neighbor $\mathbf{p}_{nn(i)}$, that is, there is no other point $\mathbf{p}_m \in P$ for which $d(\mathbf{p}_i, \mathbf{p}_{nn(i)}) = d(\mathbf{p}_i, \mathbf{p}_m)$. Then the edge $\overline{\mathbf{p}_i \mathbf{p}_{nn(i)}}$ is part of every EMST of P.*

Proof: Follows immediately from the minimality property of the EMST in Definition 4.1. ■

Theorem 4.4 implies that each point $\mathbf{p}_i$ is connected with its nearest neighbor. If there is more than one nearest neighbor with the same distance to $\mathbf{p}_i$, then $\mathbf{p}_i$ is connected with one of them in the EMST, cf. also Figure 4.2. In practice of EMST calculation, the resulting EMST in such cases depends on the numerical precision (of the processor) and/or on the order of appearance of the given points during the computation.

Theorem 4.5 (Leaves of the EMST) *Let P be a finite point set in d-dimensional space. Let $\mathbf{p}_i \in P$ be a leaf point of $EMST(P)$, that is, a point with exactly one incident edge in the EMST. Let $\mathbf{p}_j \in P$ be the point that is connected with $\mathbf{p}_i$. Then $\mathbf{p}_j$ is the nearest neighbor of $\mathbf{p}_i$.*

Proof: Follows immediately from Theorem 4.4 and the connectness of EMSTs. ■

The consequence of Theorems 4.4 and 4.5 is that each leaf point of an EMST is connected with its nearest neighbor.

The opposite of Theorem 4.4 is not true. If two points are connected by an EMST edge, the points are not necessarily the nearest neighbors of each other, as proven in Theorem 4.6.

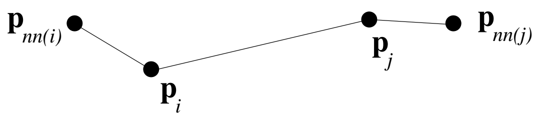


Figure 4.3: Point $\mathbf{p}_i$ is connected with $\mathbf{p}_j$ in the EMST but no point of $\{\mathbf{p}_i,\ \mathbf{p}_j\}$ is the nearest neighbor of the other one. Note that $\overline{\mathbf{p}_i\mathbf{p}_j}$ is the nearest neighbor edge between the EMST subtrees $(\{\mathbf{p}_i, \mathbf{p}_{nn(i)}\}, \{\overline{\mathbf{p}_i\mathbf{p}_{nn(i)}}\})$ and $(\{\mathbf{p}_j, \mathbf{p}_{nn(j)}\}, \{\overline{\mathbf{p}_j\mathbf{p}_{nn(j)}}\})$.

Theorem 4.6 (Property of EMST edges) *Let $\overline{\mathbf{p}_i\mathbf{p}_j}$ be an edge of the EMST of P. Then $\mathbf{p}_i$ is not necessarily the nearest neighbor of $\mathbf{p}_j$ and vice versa.*

Proof: Figure 4.3 shows a 2D configuration where a point $\mathbf{p}_i$ is connected with $\mathbf{p}_j$ but is not its nearest neighbor. ∎

Although Theorem 4.6 holds, all edges of an EMST connect subtrees of the EMST. These EMST edges represent the nearest neighbor connection between these subtrees. In fact, each edge of the EMST is in some sense a nearest neighbor edge (between subtrees) where single points can be considered as trivial subtrees of the EMST (cf. Theorem 4.7 and Algorithm 4.1).

Theorem 4.7 (Prim [Pri57]) *Let $G = (P, E)$ be a graph with weighted edges and let $\{P_1, P_2\}$ be a partition of the set P. Then there is a minimum spanning tree of G which contains the shortest among the edges with one end point in P_1 and the other in P_2.*

Proof: See [Pri57] or [PS85]. ∎

The preceding theorems show that the edges of an EMST connect points that lie close together in space. On the other hand, it can be expected for a reasonably sampled surface that the point density on the surface is higher than anywhere else in the surrounding space. In particular for non-convex surfaces and objects consisting of more than one component, points lying far apart from each other in space are unlikely to be neighboring on the surface. Furthermore, if it is necessary to reconstruct very small and detailed surface structures, the point density at those areas should be higher than at parts with less detail. With respect to these considerations the EMST turns out to be very suitable as a surface-approximating skeleton. Figure 4.1 illustrates this observation at several examples. We can observe at these examples that the EMST follows the shape of the object in a quite natural manner.

Another type of graph having the property of short edges are the Nearest-Neighbor Graphs (NNG) [Vel94]. The nearest neighbor graph of a finite point set P connects each of its points to its nearest neighbor(s). According to Theorem 4.4, the NNG is a subgraph of the EMST. A disadvantage of the NNGs is that, in contrast to the EMST, they are in general not connected.

Since we here investigate the properties of EMSTs as surface approximants it is necessary to determine the sharpest possible turn between two consecutive edges. For this purpose we calculate the minimum angle between two adjacent edges in the EMST. In order to do this we first need a consideration on the angles inside a triangle which is analyzed in the following theorem.

Theorem 4.8 (Longest edge property in a triangle) *Let t be a triangle with edges e_a, e_b, e_c where $a = l(e_a), b = l(e_b), c = l(e_c)$ are the lengths of these edges. Furthermore, let α, β, γ be the angles that are opposite to the edges e_a, e_b, e_c. We assume that $\gamma = \max(\alpha, \beta, \gamma)$ is the largest angle in the triangle. Then, $c = \max(a, b, c)$ and e_c is the longest edge in t.*

Proof: From the sine theorem [BS87] we know that

$$\frac{a}{\sin(\alpha)} = \frac{b}{\sin(\beta)} = \frac{c}{\sin(\gamma)}. \tag{4.1}$$

Additionally, we know that for $0° \leq \delta \leq 180°$, the equation

$$\sin(180° - \delta) = \sin(\delta) \tag{4.2}$$

holds.
W.l.o.g. we assume that γ with $\gamma = \max(\alpha, \beta, \gamma)$ is the largest angle. We have to show that in that case $c \geq a$ and $c \geq b$ holds. It is sufficient to show that $c \geq a$ because the sequence of equations for b will be the same. Using Equation 4.1 we have

$$c = a \cdot \frac{\sin(\gamma)}{\sin(\alpha)}.$$

Obviously, because of $180° = \alpha + \beta + \gamma$ holds, the value γ must be larger or equal to $60°$ if $\gamma = \max(\alpha, \beta, \gamma)$. After these considerations we can make a case distinction.
For $60° \leq \gamma \leq 90°$ we obtain

$$\frac{\sin(\gamma)}{\sin(\alpha)} \geq 1$$

because $\gamma \geq \alpha$, $\alpha \leq 90°$ and $\sin(\gamma) \geq \sin(\alpha)$.
In the case of $90° < \gamma \leq 180°$ we get

$$c = a \cdot \frac{\sin(\gamma)}{\sin(\alpha)} = a \cdot \frac{\sin(180° - (\alpha + \beta))}{\sin(\alpha)} = a \cdot \frac{\sin(\alpha + \beta)}{\sin(\alpha)}$$

by using Equation 4.2 and because of $(\alpha + \beta) \leq 90°$ and $\sin(\alpha + \beta) > \sin(\alpha)$. Therefore,

$$\frac{\sin(\alpha + \beta)}{\sin(\alpha)} \geq 1$$

implying $c > a$. The same cases appear for the comparison of b with c. ■

A result of Theorem 4.8 is that the EMST maximizes the angles between adjacent edges because of its minimum length property.
Using the result of the previous theorem, the minimum angle in an EMST can now be calculated.

Theorem 4.9 (Minimum angle in the EMST) *Let P be a finite point set in d-dimensional space. The minimum angle between two adjacent edges of $EMST(P)$ at an arbitrary point $\mathbf{p}_i \in P$ is $60°$.*

Proof: Let $G = (P, E) = EMST(P)$ be the EMST of a finite point set P. Consider two arbitrary edges $e_1 = \overline{\mathbf{p}_i\mathbf{p}_j}$, $e_2 = \overline{\mathbf{p}_i\mathbf{p}_k}$ of G that are incident to an arbitrary point $\mathbf{p}_i \in P$. If e_1, e_2 enclose an angle γ less than $60°$, then it is not the largest angle in the triangle $t = \triangle(\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k)$ because $180° = \alpha + \beta + \gamma$ where α, β are the angles that are opposite to e_1, e_2 in t. From Theorem 4.8 we know that the largest angle in t is opposite to the longest edge. Therefore, either e_1 or e_2 is the longest edge in t. This means that the points $\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k$ can be connected with two other edges that have a lower edge length sum than $l(e_1) + l(e_2)$. The consequence is that there also exists an edge set E' with lower edge length sum than E that connects all points of P. This is a contradiction because $G = (P, E)$ is an EMST. ■

In fact, the maximum turn of a surface that can be represented by an EMST is usually larger than 60° and depends strongly on the distribution of points which influence the length of adjacent EMST graph edges.

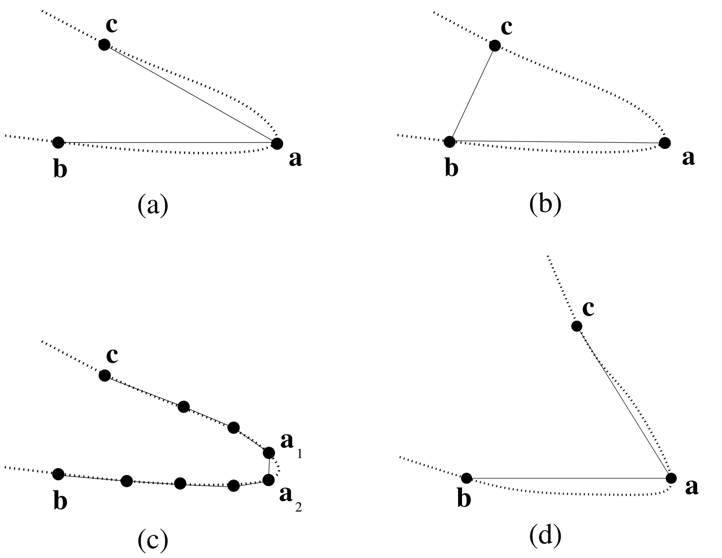


Figure 4.4: A 2D cross section through the surface of an object. The surface is drawn dotted. (a) The edges correspond to the surface structure but they are not equal to the EMST. The desired reconstruction of this surface turn would be described by the edges $\overline{\mathbf{ca}}, \overline{\mathbf{ab}}$. (b) The EMST of the points **a**,**b**,**c** connects the wrong points in comparison to the real surface. The surface is not properly sampled and the EMST of **a**, **b**, **c** does not follow the surface accurately. (c) The surface is sampled properly so that the EMST of the shown points can follow the real surface turn. (d) The angle between the two edges $\overline{\mathbf{ca}}, \overline{\mathbf{ab}}$ is high enough so that the EMST of the points **a**, **b**, **c** describes a correct surface turn.

Taking the EMST as a surface skeleton results in the fact that the properties of the EMST are also transferred to the type of surface that can be reconstructed. Therefore, the precision of the EMST as stated in Theorem 4.9 affects the distribution of points that is allowed in order to properly describe a surface by a surface graph.

A concrete example for a surface portion with a sharp edge at a surface point where the EMST precision has an effect is depicted in Figure 4.4. We see that it is crucial to scan the points at the correct areas of the surface in order to avoid wrong structures of the EMST.

Another issue which is important for computational efficiency, is the number of edges that can be incident to a point of an EMST. For the case of a point set in the plane we know the following.

Theorem 4.10 (Number of incident EMST edges to a point in 2D) *The maximum number of edges incident to a point of an EMST of a finite point set in the plane is 6.*

Proof: Consider the optimal configuration in Figure 4.5. From Theorem 4.9 we know that the smallest possible angle for an EMST is $60°$. That results in $\frac{360°}{60°} = 6$ neighbors in the plane. Now, we have to prove that all points must have the same distance to achieve the maximum number 6 for the possible connected neighbors. Consider two consecutive edges $e_1 = \overline{\mathbf{p}_i\mathbf{p}_{k_j}}$, $e_2 = \overline{\mathbf{p}_i\mathbf{p}_{k_{j+1}}}$. We know that the longest edge in a triangle through $\mathbf{p}_i, \mathbf{p}_{k_j}\mathbf{p}_{k_{j+1}}$, for example, is always opposite the largest angle in the triangle, cf. Theorem 4.8. If one of the two edges e_1, e_2 would be longer than the other one then it would be the longest edge in the whole triangle and therefore be not part of the EMST. The other edge $e_3 = \overline{\mathbf{p}_{k_j}\mathbf{p}_{k_{j+1}}}$ cannot be longer than the other two edges because the opposite angle at $\overline{\mathbf{p}_i\mathbf{p}_{k_j}}, \overline{\mathbf{p}_i\mathbf{p}_{k_{j+1}}}$

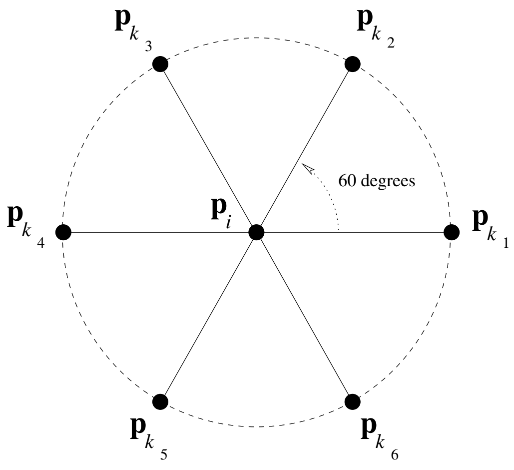


Figure 4.5: The maximum number of EMST edges around a point of the given planar point set is 6. The distances between consecutive points $\mathbf{p}_{k_i}, \mathbf{p}_{k_{i+1}}$ are equal to $d(\mathbf{p}_i, \mathbf{p}_{k_i})$ for $i \in \{1, \ldots, 6\}$. Provided that the distances to the middle point are smaller only for numerical reasons, this EMST graph is the optimal structure. On the other hand, the shown EMST graph is one of the possible EMST graphs for this point set.

is still $60°$, and at least one of the angles in the (new possible) triangle induced by edges e_1, e_2 with non-equal length would have to be larger than $60°$. Therefore, all points $\mathbf{p}_{k_j}$ must have equal distance to $\mathbf{p}_i$. ∎

In three dimensions the number of possible EMST edges around an arbitrary point in $EMST(P)$ can be higher.

Theorem 4.11 (Number of incident EMST edges to a point in 3D) *The largest vertex degree, that is, the number of incident edges of an EMST in 3D space is 12.*

Proof: We place a number of edges $e_1, \ldots, e_k$ with arbitrary length at an arbitrary point $\mathbf{p}_i$ with pairwise angles of minimum $60°$. W.l.o.g. let e_l be the shortest edge at $\mathbf{p}_i$ with scaled length 1. We place spheres of radius $\frac{1}{2}$ at $\mathbf{p}_i$ and at each so-called *anchor point* on $e_1, \ldots, e_k$ with distance 1 to $\mathbf{p}_i$. Because each pairwise angle is at minimum $60°$ the spheres cannot penetrate each other. They can only touch each other which is the case if the angle of two neighboring edges is exactly $60°$ since $\mathbf{p}_i$ and the two anchor points form an equally-sided triangle (see Figure 4.6). The number of edges with spheres at their anchor points that can be placed around $\mathbf{p}_i$ without penetrating each other corresponds to the problem of how many spheres with equal radius ($= \frac{1}{2}$) can be placed at a center sphere at $\mathbf{p}_i$. This problem is solved in [SvdW53, Lee56, CS88] and its number is limited to 12. [1]

Now, we have to show that there exists at least one configuration of edges at $\mathbf{p}_i$ which is an EMST. If all edges are replaced with edges that connect only the anchor points with $\mathbf{p}_i$ this is at least one possible configuration which is an EMST. This is because each anchor point has at least a distance of 1 to each other anchor point, the same distance as to $\mathbf{p}_i$. Therefore, the edges at $\mathbf{p}_i$ represent one of the possible EMSTs for the anchor points and $\mathbf{p}_i$. ∎

In Theorem 4.11 the maximum number of edges around a point $\mathbf{p}_i$ in the EMST has been computed. In the ideal case this value is 12 but in practice it can be expected to be much lower. Especially, if the

[1] The oldest proofs appeared in the nineteenth century [Ben74, Hop74, Gün75]. Other approaches can be found in [Boe52] and [Was78]. The whole problem area is also known as the *kissing number problem*, cf. [CS88].

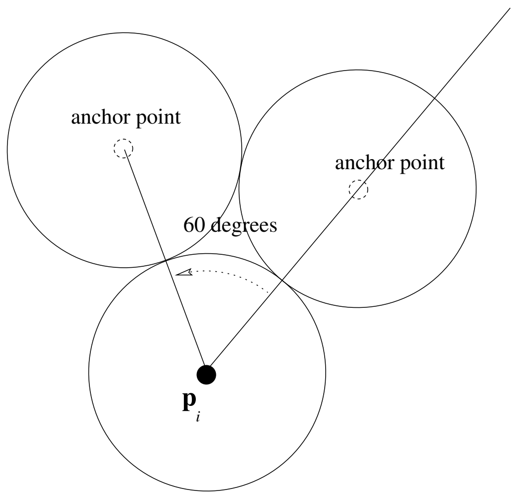


Figure 4.6: The edges that can be placed around $\mathbf{p}_i$ can have arbitrary length. At $\mathbf{p}_i$ and on each edge with distance 1 to $\mathbf{p}_i$ a sphere of radius $\frac{1}{2}$ is placed. This picture shows the situation where two edges form an angle that is exactly $60°$. All spheres are touching each other because $\mathbf{p}_i$ and the two anchor points form an equally-sided triangle.

set of points describes a two-dimensional surface in three-dimensional space, the boundary for the 2D case, which is equal to 6, applies more than for the 3D case.

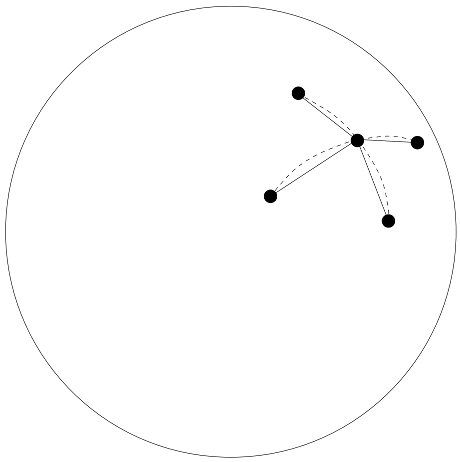


Figure 4.7: The case of four incident EMST edges (solid lines) to a point on a sphere and their corresponding surface curves (dashed lines). The nearer the points are to each other the more the situation for vertex degrees of the EMST resembles the case in the plane than the case in 3D space.

Let us consider the case of sample points that are scattered over a sphere of radius 1. If more points are present on the sphere, then the edges between the points on the sphere surface adapt better to the 3D distance between them. Therefore, the more points are scattered over the sphere the better is the correspondence of the 3D case to the 2D case. Thus, the restrictions for the 2D case apply for the number of edges around a point $\mathbf{p}_i$. An example is shown in Figure 4.7.

Table 4.1 shows the statistics of vertex degrees for the points sets of Figure 4.1. The maximum of six edges around a single point $\mathbf{p}_i$ is never reached, and even five edges around the point have never

	# of vertices of degree k in the EMST							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$k = 1$	76	325	262	167	177	83	446	145
$k = 2$	163	2020	975	379	356	211	1413	337
$k = 3$	68	287	240	139	149	73	410	133
$k = 4$	3	18	10	13	13	4	17	5
# edges	309	2649	1486	697	694	370	2285	619

Table 4.1: Statistics of vertex degrees for the EMSTs of our example point sets. The vertex degree is much lower than the theoretic upper bound of 3D EMSTs. Even the bound of 6 for the vertex degree of the 2D case is not reached.

occurred in all of our examples. The maximum value here has been four, and in average we can expect approximately two or three edges at a point.

However, for reasons of complexity the worst case has to be considered anyway. Here, if the maximum of 12 edges occurs then all surrounding points cannot have the same maximum vertex degree [SvdW53]. There are several papers on this topic in which bounds are derived from the theory of spherical packings in three-dimensional space which can be found in [CS88].

As a result, the number of EMST edges that are in average around a point can be expected equal to or less than 4.

4.2 Computational Issues

The calculation of the EMST is a classical problem of computational geometry, and efficient algorithms are known for it.

object	# points	computation times in seconds	
		$\mathrm{DT}(P)$	$EMST(P)$
torus	310	1	0.03
cup	2650	11	0.32
head	1487	6	0.18
skull	698	3	0.08
puppet	695	3	0.08
cap	371	2	0.04
pharaoh	2286	10	0.29
tori	620	1	0.07

Table 4.2: The computation times of the Delaunay triangulation and the subsequent EMST calculation for our examples in seconds on an SGI Octane R10000 at 250 MHz with 384 MByte of memory.

A frequent approach is to calculate the EMST out of the Delaunay triangulation of the point set, because the EMST is a subgraph of the Delaunay graph [PS85]. The worst case time complexity of the Delaunay triangulation is $\mathcal{O}(n^2)$ in 3D space [Joe89, Joe91]. The computation of the EMST out of a supergraph requires $\mathcal{O}(|E| \log n)$ operations, where $|E|$ denotes the number of edges in the supergraph. The number of edges in the 3D Delaunay triangulation can be $\mathcal{O}(n^2)$ in the worst case, which results in $\mathcal{O}(n^2 \log n)$ for the EMST computation out of the Delaunay triangulation. Calculation of the EMST is possible by using one of the optimal spanning tree algorithms of graph theory, for example that of Kruskal [PS85].

The computation times for our examples can be found in Table 4.2. For the calculation of the Delaunay triangulation the software of Mücke has been used [Müc93b, Müc93a, EM94].

Algorithm 4.1 Computation of $EMST(P)$

Input: *Point set* $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$.
Operation: *Compute the* $EMST$ *of* P.
Let $G_i := (\{\mathbf{p}_i\}, \emptyset)$ *be the initial one-node-trees for* $i = 1, \ldots, n$.
Let $\mathcal{G} := \{G_1, \ldots, G_n\}$ *be the set of all trees.*
while ($|\mathcal{G}| > 1$) **do**
    foreach ($G_j \in \mathcal{G}$ *with* $j \in \{ 1, \ldots, |\mathcal{G}| \}$) **do**
        Determine the nearest neighbor tree $G_k \in \mathcal{G}$ *of* G_j.
        Let $\overline{\mathbf{p}_r\mathbf{p}_s}$ *be the shortest connecting edge between* G_j *and* G_k.
        $G' := G_j \cup (\{\mathbf{p}_r, \mathbf{p}_s\}, \{\overline{\mathbf{p}_r\mathbf{p}_s}\}) \cup G_k$.
        $\mathcal{G} := (\mathcal{G} - \{G_j, G_k\}) \cup \{G'\}$.
    end
end
Output: $EMST(P)$.

Another possibility is to calculate the EMST directly. A particularly suited spanning tree algorithm for that purpose is that of Borůvka. According to Ottmann & Widmayer [OW90] the algorithm of Borůvka [Bor26] is reported to be the historically first algorithm for the computation of a minimum spanning tree. It is shown in Algorithm 4.1. The union of graphs $G_1 = (P_1, E_2)$ and $G_2 = (P_2, E_2)$ used in the algorithm is defined as $G := (P_1 \cup P_2, E_1 \cup E_2)$. At the beginning of the algorithm, an initial forest of one-node-trees is established. Each vertex of P defines one tree of that forest. Then for each tree G_j its *nearest neighbor tree* G_k is computed, which is the tree that contains the point $\mathbf{p}_s$ not in G_j which has the shortest Euclidean distance to the nearest point $\mathbf{p}_r$ of G_j. Both trees are then merged into one single tree, so that the number of all trees is decreased by one. This tree merging process is repeated until the set of these trees has reduced to one tree.

Prim's algorithm [Pri57] is a special case of the algorithm of Borůvka. There, the process starts with a single tree defined by an arbitrary vertex node. This initial tree is then sequentially extended by its nearest neighbor connection to a new vertex, that is not already part of the current tree. As before, the process is finished if all vertices have been connected to one tree.

4.3 Discussion

This chapter has dealt with the properties which make the EMST useful as an initial skeleton of a surface. Experimental examples have demonstrated the favorable behavior of the EMST. Furthermore, algorithmic aspects concerning the calculation of the EMST have been treated.

Chapter 5

Environment Graphs

In the previous chapter we have seen that the EMST can serve as a good surface approximation. However, in Figure 5.1 we can notice that the torus ring is not closed with edges because of the minimum length property of the EMST (cf. Chapter 4). As consequence, in this chapter we consider more dense graph schemes, the so-called environment graphs which extend the concept of the EMST and preserve to a large extent the favorable surface approximation properties of the edges which we could observe for the EMST. Furthermore, we cope with the problem of bridge edges which may connect parts of a surface over a large distance in an undesirable manner, as could be noticed for example for the double torus in the preceding chapter, cf. Figure 4.1.

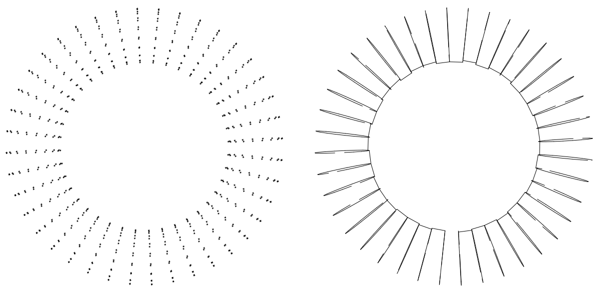


Figure 5.1: The point set and the EMST of a torus that consists of rings of points.

5.1 Environment Graphs

The following definition describes a very general concept of environment graphs which can be specialized to several known classes of graphs.

Definition 5.1 (Environment graph (EG)) *Let $\mathcal{U}$ be a function that assigns to every line segment s in the d-dimensional space $\mathbb{R}^d$ a set $\mathcal{U}(s)$ of environments of s. The **$\mathcal{U}$-edge-environment graph ($\mathcal{U}$-EG)** of a finite set of points in $\mathbb{R}^d$ is a graph with vertex set P, whose edge set consists of all those line segments e induced by pairs of points in P for which there exists an $U(e) \in \mathcal{U}(e)$ with $U(e) \cap P = \emptyset$, that is, the U-environment of e does not contain any point of P.*

A typical example of an EG is the Delaunay graph whose edges are just those line segments of its vertex set which have an empty surrounding sphere [PS85, Ede87]. In that case, $\mathcal{U}(s)$ is the set of all surrounding spheres of s.

Definition 5.2 (Euclidean minimum spanning environment (EMSE)) *For any pair of points* $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$*, the* ***Euclidean minimum spanning environment (EMSE)*** *of their connecting line segment* s *is the intersection of the closed ball touching* $\mathbf{q}$ *and centered at* $\mathbf{p}$*, with the closed ball touching* $\mathbf{p}$ *and centered at* $\mathbf{q}$*. The EMSE is denoted by* $U_{EMSE}(s)$*.*
Let P *be a set of points in* $\mathbb{R}^d$*. The EMSE-graph of* P *is the* $\mathcal{U}$*-EG with* $\mathcal{U}(s) := \{U_{EMSE}(s)\}$ *for every line segment* s*.*

The reason for introducing the EMSE lies in the following theorem.

Theorem 5.3 (EMSE-property) *Let* $\mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathbb{R}^d$*,* s *the line segment connecting* $\mathbf{p}$ *and* $\mathbf{q}$*.*

(1) s *belongs to an Euclidean minimum spanning tree of* $Q = \{\mathbf{p}, \mathbf{q}, \mathbf{r}\}$ *if and only if* $\mathbf{r} \notin U_{EMSE}(s)$*.*

(2) For any finite set P *of points in* $\mathbb{R}^d$*, the EMSTs of* P *are subgraphs of the EMSE-graph of* P*.*

Proof:

(1) By definition of $U_{EMSE}(s)$ there is no point inside the spheres of radius $d(\mathbf{p}, \mathbf{q})$ around **p**,**q** so that one of the points **p**,**q** is the nearest neighbor of the other one in Q. By Theorem 4.4 we know that nearest neighbors are part of an EMST.

(2) Each line segment s of the $EMST(P)$ with $s = \overline{\mathbf{pq}}$ does not have a point **r** inside (and not on the border of) the intersection of the spheres around $\mathbf{p}, \mathbf{q}$ with radius $d(\mathbf{p}, \mathbf{q})$ which is the condition for each edge of the EMSE-graph. Let us assume such a point **r** would exist for $\mathbf{p}, \mathbf{q}$. W.l.o.g. there exists a path from **p** (or **q**) to **r** that does not contain the point **q** (or **p**). Obviously, we know that $d(\mathbf{q}, \mathbf{r}) < d(\mathbf{p}, \mathbf{q})$. If we modify the $EMST(P)$ by removing $\overline{\mathbf{pq}}$ and adding the edge $\overline{\mathbf{qr}}$ we get a connected graph with lower edge length sum. This cannot be true because the $EMST(P)$ is of minimal length.

■

The EMSE-graph in general has more edges than an EMST which, according to the previous theorem, is in some sense a "lower bound" of the EMSE-graph. An "upper bound" is given by the next theorem.

Theorem 5.4 (Delaunay property of the EMSE-graph) *Let* P *be a finite set of points in* $\mathbb{R}^d$*. The EMSE-graph of* P *is a subgraph of the Delaunay graph of* P*.*

Proof: The EMSE of a line segment s comprehends an empty surrounding sphere of s. ■

The following corollary is an immediate consequence of this theorem.

Corollary 5.5 (Planarity of the EMSE-graph) *Let* P *be a finite set of points in the plane. The points are assumed in general position, that is, no four of them are co-cyclic. Then the EMSE-graph of* P *is a planar graph.*

Proof: The Delaunay graph of points in general position is a planar graph. Because the EMSE-graph is a subgraph of the Delaunay graph, the EMSE-graph is planar, too. ■

Another property of the EMSE-graph which is an immediate consequence of the EMSE-property is stated in Theorem 5.6.

Theorem 5.6 (Cycle length of the EMSE-graph) *If a cycle of the EMSE-graph has length 3 then its vertices define an equal-sided triangle.*

Proof: The only case that all edges of a triangle have an empty EMSE-environment is the equal-sided triangle. ■

An implication of the theorem is that EMSE-graphs should be triangle-free with high probability. Several properties of the EMST which are of local nature can also be found for the EMSE-graph.

Theorem 5.7 (Local EMSE-graph properties) *Let P be a finite point set in $\mathbb{R}^d$ and $\mathbf{p}, \mathbf{q} \in P$.*

(1) Let $\mathbf{p}$ be a leaf point of the EMSE-graph of P, that is, a point with exactly one incident edge, which is connected with a point $\mathbf{q}$. Then $\mathbf{q}$ is the nearest neighbor of $\mathbf{p}$.

(2) Let $\overline{\mathbf{pq}}$ be an edge of the EMSE-graph of P. Then $\mathbf{p}$ is not necessarily the nearest neighbor of $\mathbf{q}$ and vice versa.

(3) The minimum angle between two adjacent EMSE-graph edges at an arbitrary point $\mathbf{p}$ of the graph is $60°$.

(4) The maximum number of edges incident to a point of an EMSE-graph of a point set in the plane is 6.

(5) The largest vertex degree, that is, the number of incident edges, of an EMSE-graph in 3D space is 12.

Proof: Analogously to the EMST case, cf. Chapter 4. ■

The suitability of the EMSE-graph as a framework of surface interpolation lies in the fact that the EMSE is not able to "escape" into the empty space as the surrounding spheres of the Delaunay triangulations can. Figures 5.2 and 5.3, left column, show the Delaunay triangulation of our series of sample data sets. In the second column from the left, the corresponding EMSE-graphs are depicted.

Here the question arises for environment graphs induced by spheres which are prevented to escape. An immediate possibility is that the corresponding environment of an edge is bounded by the unique sphere having the edge as its diameter. This type of graph is well-known as *Gabriel graph* [GS69] which has been preferably used in the plane. Both graph classes, the EMSE-graphs and the Gabriel graphs, can be seen in the much wider framework given by the concept of β-environment graphs.

Definition 5.8 (β-environment graph (β-EG)) *Let $e = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ be a line segment, and H a plane in $\mathbb{R}^3$ containing e.*

For $\beta \geq 0$, let $\mathbf{p}'$, $\mathbf{q}'$ be points on the line induced by e located symmetrically with respect to its center $\mathbf{m} := \frac{1}{2}(\mathbf{p} + \mathbf{q})$, that is

$$\mathbf{p}' = \beta\mathbf{p} + (1 - \beta)\mathbf{m},$$
$$\mathbf{q}' = \beta\mathbf{q} + (1 - \beta)\mathbf{m}.$$

Let $E'_\beta(e)$ be the intersection of the disc in H with center $\mathbf{p}'$ through $\mathbf{q}$ with the disc in H with center $\mathbf{q}'$ through $\mathbf{p}$, cf. Figure 5.4 (left).

For $\beta < 0$, let $\mathbf{p}'$, $\mathbf{q}'$ be points on the perpendicular line in H through the center $\mathbf{m}$ of an edge $e = \overline{\mathbf{pq}}$ located symmetrically with respect to its center, with

$$||\mathbf{p}' - \mathbf{m}|| = ||\mathbf{q}' - \mathbf{m}|| = -\beta \cdot ||\mathbf{q} - \mathbf{p}||,$$

where $\| \cdot \|$ denotes the Euclidean norm. The environment $E'_\beta(e)$ is defined as the intersection of the disc with center $\mathbf{p}'$ through $\mathbf{p}$ and $\mathbf{q}$ with the disc with center $\mathbf{q}'$ through $\mathbf{p}$ and $\mathbf{q}$, cf. Figure 5.4 (right).

*Let $E_\beta(e)$ be the set of points obtained by rotating $E'_\beta(e)$ around e. $E_\beta(e)$ is called the **β-environment of** e. The environment graph with β-environments as environments is called **β-environment graph**. In 2D space H is replaced with the 2D space and $E_\beta(e) := E'_\beta(e)$.*

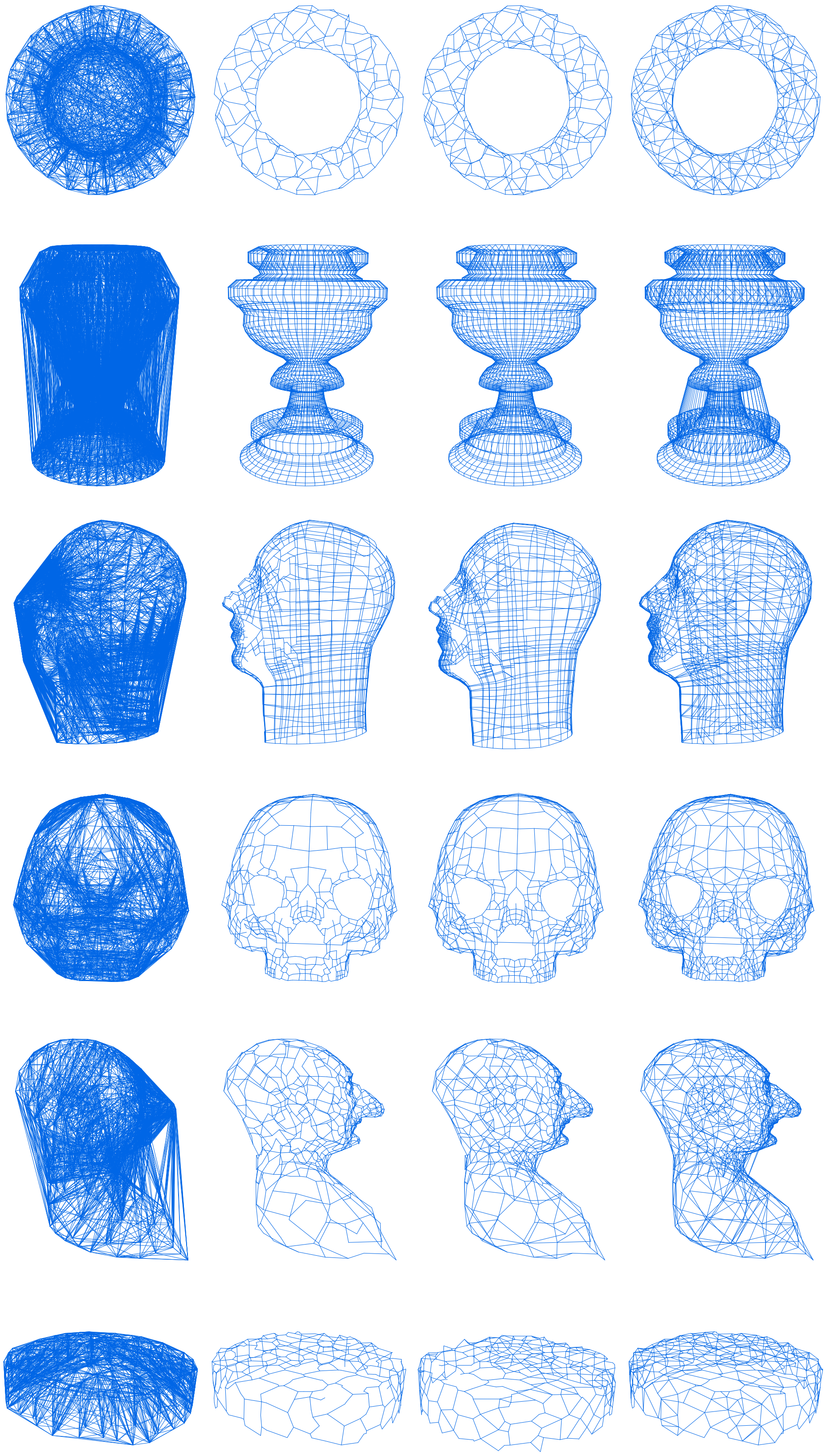


Figure 5.2: From left to right: the Delaunay triangulation $DT(P)$, the β-environment graph for $\beta = 1, \frac{1}{2}, 0$. Note, that the EMSE-graph corresponds to a β-environment graph with $\beta = 1$.

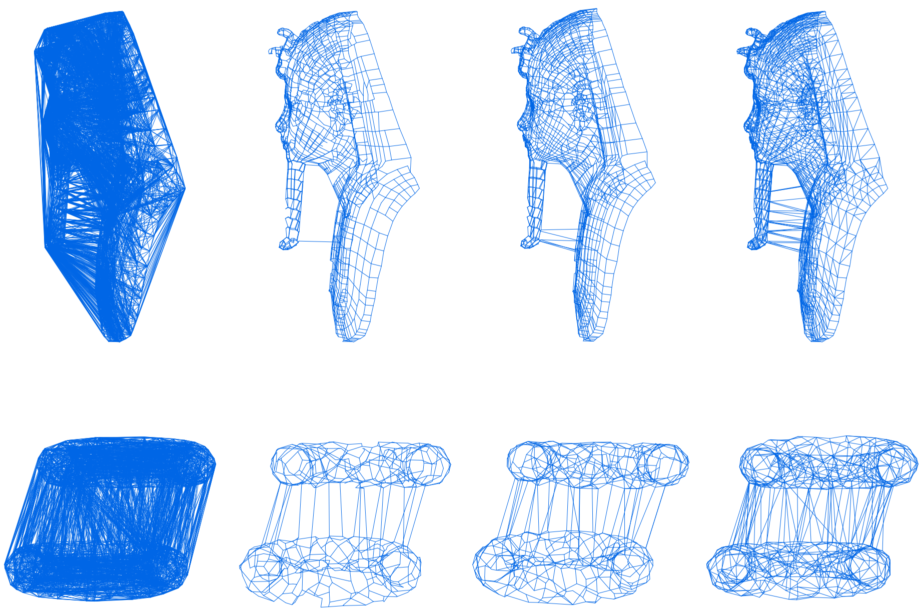


Figure 5.3: From left to right: the Delaunay triangulation $DT(P)$, the β-environment graph for $\beta = 1, \frac{1}{2}, 0$. Note, that the EMSE-graph corresponds to a β-environment graph with $\beta = 1$.

Sometimes we also will use $E_\beta(\mathbf{p}, \mathbf{q}) = E_\beta(e)$ as another notation of the β-environment of an edge $e = \overline{\mathbf{pq}}$.

In 2D space, the value $\beta = 0$ yields the Gabriel graph, in 2D and 3D space $\beta = 1$ yields the EMSE-graph.

In two dimensions, the β-environment graphs for $\beta \geq 0$ correspond to the *lune-based neighborhood graphs* of Kirkpatrick and Radke for their $\beta \geq 1$, for $\beta \leq 0$ to their *circle-based neighborhood graphs* for their β between 0 and 1 [KR85, Rad88]. Recently, Rao [Rao98] has independently generalized the neighborhood graphs to higher dimensions. Up to the β-parameterization, our β-environment graphs are the same as his lune-based neighborhood graphs.

Figures 5.2 and 5.3 show β-EGs for $\beta = 1$, $\frac{1}{2}$, and 0 of our sample data sets. We can notice that also in the 3D-case the resulting graph yields a reasonable, almost planar approximation. The number of "crossing" edges is very small, and decreases with increasing β. Figure 5.5 shows the edges of β-environment graphs for $\beta = -\frac{1}{2}$ which additionally are Delaunay edges. Even with this restriction, many edges useless for reconstruction are delivered. On the other hand, the chance seems to be good that the desired mesh is a subset of this graph. Based on this observation it might be possible to find a surface reconstruction algorithm that works by eliminating the superfluous edges from the graph.

As we can see from our examples, the number of edges in the β-EG decreases with increasing β. Some statistics of vertex degrees, that is, the number of points with k incident edges in the β-EGs for $\beta = 1, \frac{1}{2}, 0$ are shown in Tables 5.1, 5.2, 5.3.

An interesting observation is that β-environment graphs have a hierarchical structure.

Theorem 5.9 (Subgraph hierarchy of the β-EG) *For a given finite point set P in 3D space, if $\beta_1 > \beta_2$, the β_1-EG of P is subgraph of the β_2-EG of P.*

Proof: The property results from the observation that $E_{\beta_2}(\mathbf{p}, \mathbf{q}) \subset E_{\beta_1}(\mathbf{p}, \mathbf{q})$. ■

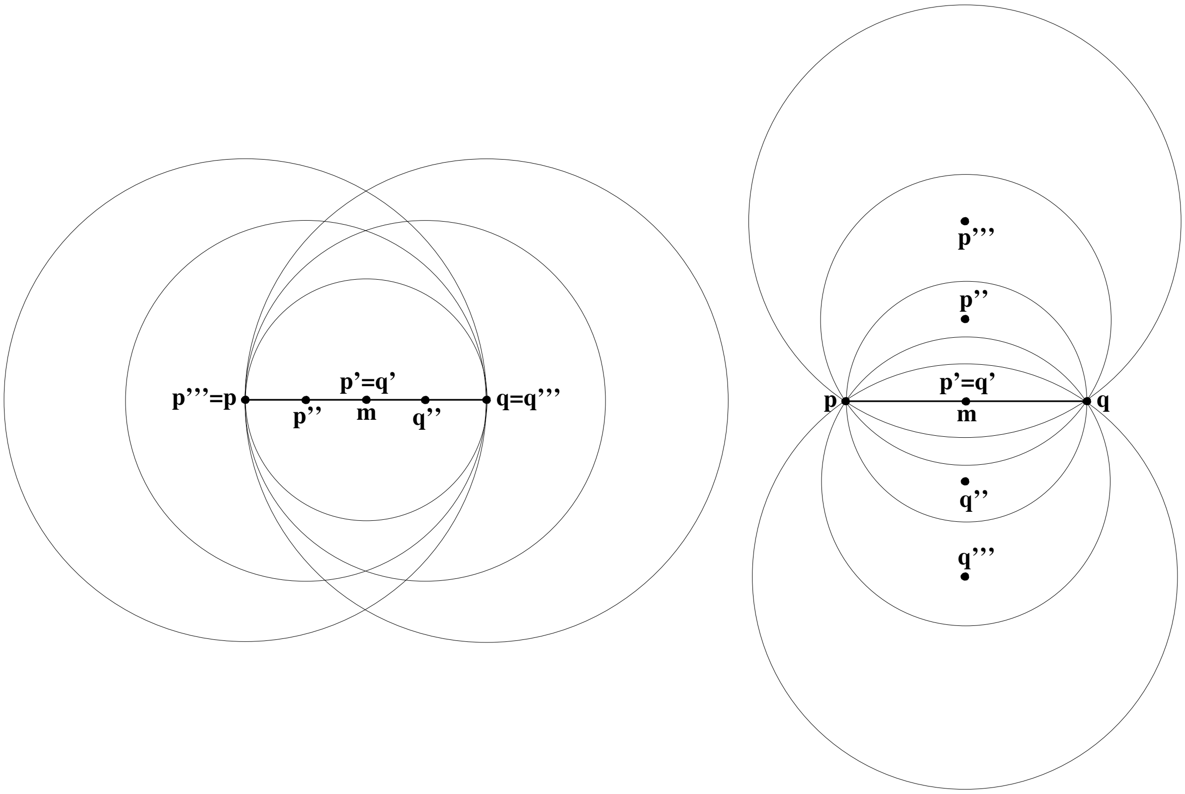


Figure 5.4: Left: The β-environment of an edge $\overline{\mathbf{pq}}$ for $\beta \geq 0$ in the plane. The points for the values of $\beta = 0$ ($\mathbf{p}', \mathbf{q}'$), $\beta = \frac{1}{2}$ ($\mathbf{p}'', \mathbf{q}''$), $\beta = 1$ ($\mathbf{p}''', \mathbf{q}'''$). Right: The β-environment for $\beta \leq 0$ in the plane. The points for the values of $\beta = 0$ ($\mathbf{p}', \mathbf{q}'$) and two values β_1 ($\mathbf{p}'', \mathbf{q}''$), β_2 ($\mathbf{p}''', \mathbf{q}'''$) with $\beta_2 < \beta_1$.

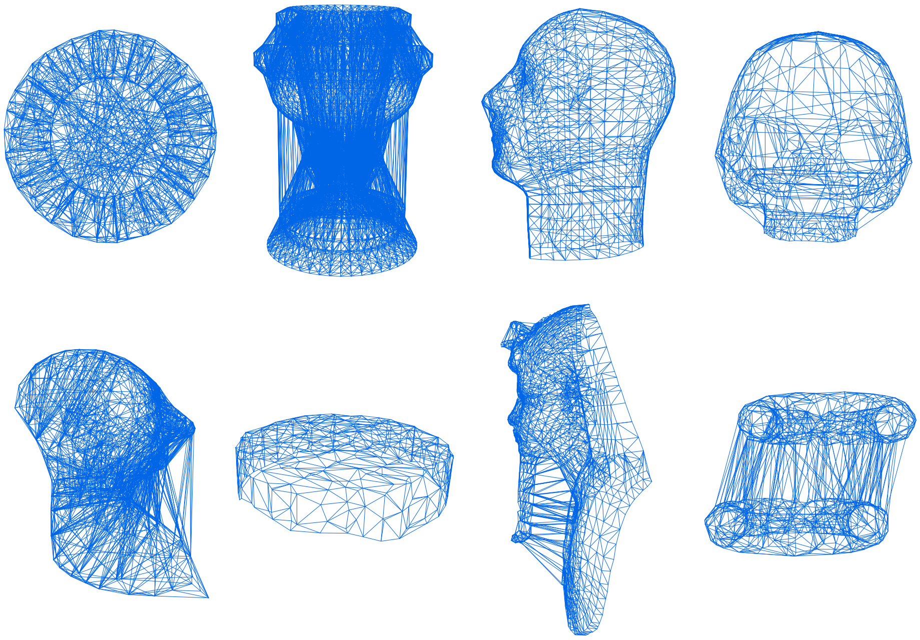


Figure 5.5: The β-environment graph restricted to the edges of the Delaunay triangulation, for $\beta = -\frac{1}{2}$.

	# of vertices of degree k in the 1-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$k = 1$	5	0	18	15	8	12	10	8
$k = 2$	100	0	253	193	179	154	439	190
$k = 3$	179	50	482	355	400	184	1111	345
$k = 4$	26	2550	734	133	108	21	725	72
$k = 5$	0	50	0	2	0	0	1	5
# edges	426	5300	2453	1004	999	484	3563	868

Table 5.1: The statistic of vertex degrees for $\beta = 1$.

	# of vertices of degree k in the $\frac{1}{2}$-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$k = 1$	0	0	4	2	0	0	1	0
$k = 2$	42	0	108	50	52	83	144	82
$k = 3$	162	50	384	320	272	183	767	307
$k = 4$	92	2450	918	295	294	92	1080	174
$k = 5$	14	150	66	30	74	13	268	49
$k = 6$	0	0	7	1	3	0	26	8
# edges	504	5350	2708	1199	1242	574	4203	1037

Table 5.2: The statistic of vertex degrees for $\beta = \frac{1}{2}$.

In the following we will consider some useful properties of β-EGs.

Theorem 5.10 (Properties of the β-EG, $\beta \geq 0$) *β-environment graphs for $\beta \geq 0$ have the following properties:*

(1) The environment $E_\beta(\mathbf{p}, \mathbf{q})$ is the intersection of the ball with center $\mathbf{p}'$ through $\mathbf{p}$ and $\mathbf{q}$ with the ball with center $\mathbf{q}'$ through $\mathbf{p}$ and $\mathbf{q}$.

(2) The β-EG is a subgraph of the Delaunay graph.

(3) For a set of points in the plane in general position, the β-EG is a planar graph.

(4) If $0 \leq \beta \leq 1$ then the β-EG is a connected graph.

Proof:

(1) By definition, $E'_\beta(\mathbf{p}, \mathbf{q})$ is the intersection of the discs in E with centers $\mathbf{p}', \mathbf{q}'$ and radii $d(\mathbf{p}', \mathbf{q})$, $d(\mathbf{p}, \mathbf{q}')$ which is rotated around $e = \overline{\mathbf{pq}}$. If we first rotate the discs around $\mathbf{p}', \mathbf{q}'$ we get balls around the points through $\mathbf{p}, \mathbf{q}$. The intersection of these balls is $E_\beta(\mathbf{p}, \mathbf{q})$.

(2) For $\beta \geq 0$ the sphere around $\mathbf{m}$ with radius $\frac{1}{2} \cdot d(\mathbf{p}, \mathbf{q})$ is obviously contained in $E_\beta(\mathbf{p}, \mathbf{q})$. It is known from the Delaunay graph, that all Delaunay edges have at least one circumscribing empty sphere.

(3) The β-EG is a planar graph because it is a subgraph of the Delaunay graph which is known to be a planar graph for points in general position.

	# of vertices of degree k in the 0-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$k=1$	0	0	0	0	0	0	0	0
$k=2$	7	0	18	8	1	31	11	11
$k=3$	59	1	101	83	31	100	167	116
$k=4$	118	1472	377	265	221	127	551	221
$k=5$	84	773	548	226	255	87	733	148
$k=6$	34	302	416	100	146	19	682	75
$k=7$	7	100	25	13	29	7	110	32
$k=8$	1	2	2	3	11	0	27	8
$k=9$	0	0	0	0	1	0	4	8
$k=10$	0	0	0	0	0	0	0	1
$k=11$	0	0	0	0	0	0	0	0
$k=12$	0	0	0	0	0	0	1	0
# edges	672	6142	3637	1585	1715	734	5759	1407

Table 5.3: The statistic of vertex degrees for $\beta = 0$.

(4) From Theorem 5.9 we know that the 1-EG is a subgraph of every β-EG with $0 \le \beta \le 1$. From Theorem 5.3 we know that the EMST is a subgraph of the EMSE-graph, which is identical to the 1-EG. Because the EMST is connected, the 1-EG and thus all β-EGs under consideration are connected.

■

Theorem 5.11 (Properties of the β-EG, $\beta < 0$) *β-environment graphs for $\beta < 0$ have the following properties:*

(1) The β-EG generally is not a subgraph of the Delaunay graph.

(2) For a vertex set in the plane, the β-EG generally is not a planar graph.

(3) Let $\mathbf{r}$ be a point in $E_\beta(\mathbf{p},\mathbf{q})$. Then the angle γ at $\mathbf{r}$ of the line segments $\overline{\mathbf{rp}}$ and $\overline{\mathbf{rq}}$ satisfies $\gamma \ge \gamma(\beta) := \arcsin(\frac{1}{\sqrt{1+4\beta^2}})$, $90° \le \gamma(\beta) < 180°$, with equality for the points on the boundary of $E_\beta(\mathbf{p},\mathbf{q})$.

(4) Let $\mathbf{r}$ be a point with an angle at the line segments $\overline{\mathbf{rp}}$ and $\overline{\mathbf{rq}}$ at $\mathbf{r}$ which is at least equal to γ, $90° \le \gamma < 180°$. Then $\mathbf{r}$ is in $E_{\beta(\gamma)}(\mathbf{p},\mathbf{q})$ with $\beta(\gamma) := (\cot\gamma)/2$.

Proof:

(1) As known for the Delaunay graph every edge has at least one circumscribing empty sphere. Because of Theorem 5.9, for every edge where a point **r** is inside its smallest circumscribing sphere a value smaller than 0 for β' can be found so that $\mathbf{r} \in E'_\beta(\mathbf{p},\mathbf{q})$.

(2) Non-planar means that there can be intersections in the plane. Figure 5.6 shows an example of a β-EG, $\beta < 0$, in the plane with intersecting edges.

(3) We consider $E'_\beta(\mathbf{p},\mathbf{q})$ in the plane H spanned by $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$, cf. Figure 5.7. For $\beta < 0$, the planar environment $E'_\beta(\mathbf{p},\mathbf{q})$ consists of two disc segments of identical shape, separated by the

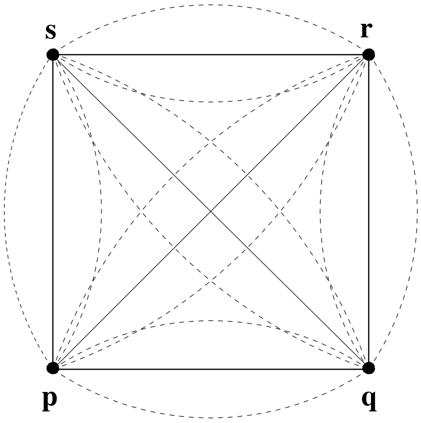


Figure 5.6: A β-EG for $\beta < 0$ in the plane. It shows that intersecting line segments are possible. The smaller β gets the more likely intersections become.

line segment $e = \overline{\mathbf{pq}}$. The definitions yield a radius of

$$r = \frac{\sqrt{1+4\beta^2}}{2} \cdot ||e||.$$

By the sine-theorem, the ratio of $||e||$ and the sine of the angle γ which is opposite to e in the triangle $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$, is equal to $2r$, that is

$$\frac{||e||}{\sin(\gamma)} = \sqrt{1+4\beta^2} \cdot ||e||,$$

if $\mathbf{r}$ is on the circular boundary of the disc segment. This is just the formula of the theorem written reciprocally. If $\mathbf{r}$ is in the inner of the disc segment, γ is larger than on the boundary. Because $\gamma > 90°$, that means that $\sin(\gamma)$ is less than for the boundary, and thus the inequality of the theorem holds.

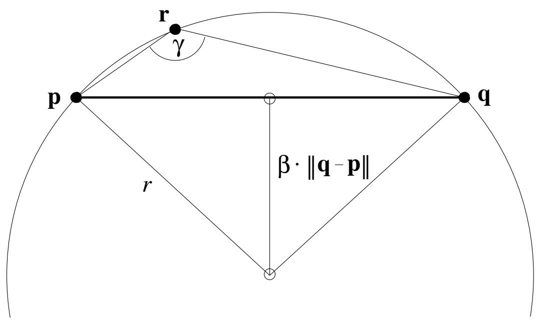


Figure 5.7: The angles at points inside the β-environments.

(4) From (3) we know that the angle is equal to γ if $\mathbf{r}$ is on the arc bounding $E_{\beta(\gamma)}(\mathbf{p}, \mathbf{q})$. Figure 5.8 depicts the situation if $\mathbf{r}$ is outside of $E_{\beta(\gamma)}(\mathbf{p}, \mathbf{q})$ and an edge of the triangle induced by $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ intersects the arc. In this case we get

$$180° = \gamma + \alpha + \delta = \varphi + (\alpha + \alpha') + \delta,$$

that is,

$$\varphi = \gamma - \alpha'.$$

Because $\alpha' > 0$, the angle at $\mathbf{r}$ is less than γ.

If the edges of the triangle do not intersect the arc, it can be immediately seen, that the angle at $\mathbf{r}$ is larger than γ.

Thus no points outside of $E_{\beta(\gamma)}(\mathbf{p}, \mathbf{q})$ with angle equal or larger than γ exist.

■

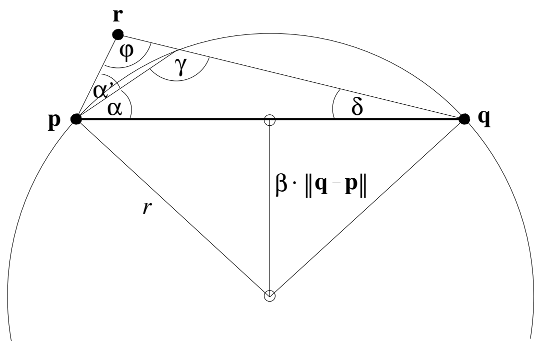


Figure 5.8: The angle at points that are outside the β-environment.

Corollary 5.12 (Angles at distant points) *Let $\mathbf{r}$ be a point outside the environment $E_\beta(\mathbf{p}, \mathbf{q})$ of an edge $e = \overline{\mathbf{pq}}$, $\beta \leq 0$. Then the angle at $\mathbf{r}$ is less than the angle γ satisfying $\sin(\gamma) \leq \frac{1}{\sqrt{1+4\beta^2}}$, $\gamma \geq 90°$.*

Proof: The corollary is an immediate consequence of assertion (4) of the preceding Theorem 5.11.

An interesting observation for later use is that for a triangular mesh which is a subgraph of a β-EG, for $\beta < 0$, the maximum angle of a triangle is less than the angle γ corresponding to β.

5.2 Clustered Environment Graphs

As we have recognized from the examples, EMSTs and environment graphs are in principle well-suited for reconstruction. They, however, show two difficulties which may sometimes occur. The first difficulty is the occurrence of long "bridge edges" which could already be noticed in some of the examples of the previous chapter. In order to avoid bridge edges, we introduce the clustered environment graphs which are subgraphs of the environment graphs.

Another effect is that the environment graph of a surface may not be locally planar, even in the clustered version. In the plane, for example, this may happen for co-cyclic points. We give heuristic criteria in order to identify critical edges and to remove them from the graph.

5.2.1 Intersecting Edges

We can notice in the examples that the edges of the environment graphs seem to be located close to the surface without "intersection".

In order to reduce the unlikely case of intersecting edges further, we use the following concept of χ-intersecting line segments generalizing the notion of intersection of line segments from the plane to line segments in space.
This concept uses the so-called *dihedral angle* between adjacent triangles.

Definition 5.13 (Dihedral angle) *Let t_1 and t_2 be two triangles which share a common edge $\overline{\mathbf{pq}}$. Then the intersection of t_1 and t_2 with a plane perpendicular to, and intersecting $\overline{\mathbf{pq}}$ consists of two incident line segments. The smaller one of the two angles between those line segments is denoted as the* ***dihedral angle*** *between t_1 and t_2.*

Now, χ-intersecting line segments can be defined as follows.

Definition 5.14 (χ-intersecting line segments) *Let be $\chi > 0°$. Two line segments $s_1 = \overline{\mathbf{p}_1\mathbf{q}_1}$ and $s_2 = \overline{\mathbf{p}_2\mathbf{q}_2}$ in 3D space without common vertex are called* ***χ-intersecting*** *if they have the following properties:*

(1) For any two triangles $t_1 = \triangle(\mathbf{p}_1, \mathbf{q}_1, \mathbf{r}_1)$ and $t_2 = \triangle(\mathbf{p}_2, \mathbf{q}_2, \mathbf{r}_2)$, $\mathbf{r}_1 \in \{\mathbf{p}_2, \mathbf{q}_2\}$, $\mathbf{r}_2 \in \{\mathbf{p}_1, \mathbf{q}_1\}$, which share a common edge, the dihedral angle between t_1 and t_2 is less than χ.

(2) The dihedral angles at $\overline{\mathbf{p}_1\mathbf{q}_1}$ and $\overline{\mathbf{p}_2\mathbf{q}_2}$ exceed $90°$.

The following theorem shows that the concept of χ-intersecting edges is reasonable in the plane.

Theorem 5.15 *Two line segments $s_i = \overline{\mathbf{p}_i\mathbf{q}_i}$, $i = 1, 2$, in the plane intersect if and only if the s_i are χ-intersecting for $\chi = 0°$.*

Proof: If s_1 and s_2 intersect, all the pairs of triangles mentioned in (1) of the definition have a dihedral angle of $0°$. This implies $\chi = 0°$. The angles of (2) are equal to $180°$ and thus larger than $90°$, as demanded.
If s_1 and s_2 do not intersect, we distinguish between two cases. The first one is that each of the segments is completely in one of the half-planes induced by the line of the other. Then the triangles obtained by triangulating the quadrilateral spanned by s_1 and s_2 have a dihedral angle of $180°$. Otherwise, the line of one of the segments, say s_1, intersects the other segment, here s_2. Let $\mathbf{p}_1$ be the vertex of s_1 that is closer to s_2. Then the triangles $\triangle(\mathbf{p}_2, \mathbf{q}_2, \mathbf{p}_1)$ and $\triangle(\mathbf{p}_1, \mathbf{q}_1, \mathbf{p}_2)$ have a dihedral angle of $180°$. In both cases the condition of the theorem does not hold. ■

Our experimental experience shows that $60° \leq \chi \leq 90°$ is a suitable choice in the case of curved surfaces. The examples of this chapter have been constructed with a value of $\chi = 90°$.

5.2.2 Clustered β-Environment Graphs

Clustered β-environment graphs are obtained by a greedy clustering scheme which yields a subgraph of a β-environment graph without edges of exceptional length.
The greedy clustering scheme consists of three phases:

1. For every point $\mathbf{p}$, a next nearest neighbor $\mathbf{q}$ of $\mathbf{p}$ is iteratively determined in order of increasing distance, as long as the edge $\overline{\mathbf{pq}}$ belongs to the β-environment graph. Each of these edges $\overline{\mathbf{pq}}$ is stored as virtually incident to $\mathbf{p}$. That approach corresponds to an iterative nearest neighbor extension of the neighborhood of $\mathbf{p}$, under consideration of the empty-β-environment constraint.

   For every point $\mathbf{p}$, a radius $r(\mathbf{p})$ is defined. $r(\mathbf{p})$ is the Euclidean length of the last edge $\overline{\mathbf{pq}}$ that was added as virtually incident to $\mathbf{p}$.

Algorithm 5.1 Computation of the clustered β-environment graph

Input: *Point set* $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ *in 3D space.*
Operation: *Calculation of the clustered* β*-environment graph:*
$G := (P, \emptyset)$.
$H := (P, \emptyset)$.
$r(\mathbf{p}) := 0$ *for all* $\mathbf{p} \in P$.
foreach ($\mathbf{p} \in P$) **do**
  repeat
    Search the next nearest neighbor **q** *of* **p**.
    if (*the* β*-environment of edge* $\overline{\mathbf{pq}}$ *is empty*) **then**
      Store this edge $\overline{\mathbf{pq}}$ *as* ***virtually incident*** *to* **p**.
      $r(\mathbf{p}) := \max(r(\mathbf{p}), d(\mathbf{p}, \mathbf{q}))$. // update sphere radius
    end
  until ($\overline{\mathbf{pq}}$ *does not have an empty* β*-environment*)
end
Insert all ***virtual edges*** *into* H.
repeat
  foreach (*pair* $\mathbf{p}, \mathbf{q} \in P$ *with intersecting spheres, that is, where* $d(\mathbf{p}, \mathbf{q}) \leq r(\mathbf{p}) + r(\mathbf{q})$) **do**
    if (*the connection* $\overline{\mathbf{pq}}$ *has an empty* β*-environment*) **then**
      Store $\overline{\mathbf{pq}}$ *as* ***virtually incident*** *to* **p** *and* **q**.
      $r(\mathbf{p}) := \max(r(\mathbf{p}), d(\mathbf{p}, \mathbf{q}))$. // update sphere radii
      $r(\mathbf{q}) := \max(r(\mathbf{q}), d(\mathbf{p}, \mathbf{q}))$.
    end
    if ($r(\mathbf{p}) > d(\mathbf{p}, \mathbf{q})$) **then** $r(\mathbf{q}) := \max(r(\mathbf{q}), r(\mathbf{p}) - d(\mathbf{p}, \mathbf{q}))$;
    if ($r(\mathbf{q}) > d(\mathbf{p}, \mathbf{q})$) **then** $r(\mathbf{p}) := \max(r(\mathbf{p}), r(\mathbf{q}) - d(\mathbf{p}, \mathbf{q}))$;
  end
  Insert all ***new virtual edges*** *into* H.
until (*no edge has been memorized as* ***virtually incident*** *and no* $r(\mathbf{p}), r(\mathbf{q})$ *has been updated*)
foreach (*virtual edge* $\overline{\mathbf{pq}}$ *of* H *in order of increasing length*) **do**
  if (*the connection* $\overline{\mathbf{pq}}$ *does not* χ*-intersect with all edges of* G) **then**
    Insert $\overline{\mathbf{pq}}$ *into* G.
  end
end
Output: G *as clustered* β*-environment graph of* P.

2. All those edges $\overline{\mathbf{pq}}$ with empty β-environment are added as virtual edges for which $d(\mathbf{p}, \mathbf{q}) \leq r(\mathbf{p}) + r(\mathbf{q})$.

   All radii $r(\mathbf{p})$ are updated with the maximum length of the virtual edges incident to **p**.

   For all **q** with $d(\mathbf{p}, \mathbf{q}) < r(\mathbf{p})$, the radius is updated to $r(\mathbf{q}) := \max\{r(\mathbf{q}), r(\mathbf{p}) - d(\mathbf{p}, \mathbf{q})\}$.

   Step 2 is repeated until these actions do not deliver any new virtual edge and no update of any radius occurs.

3. Then, all virtual edges are considered in order of increasing length. If a virtual edge $\overline{\mathbf{pq}}$ does not χ-intersect any edge of the current graph G, then it is added to G. This process is terminated if all virtual edges have been considered.

The motivation for this clustering approach is as follows. Step 1 of the algorithm determines an intial radius of a sphere for each point **p** in which all connecting edges $\overline{\mathbf{pq}}$ to points $\mathbf{q} \in P$ inside this sphere have an empty β-environment. This radius is a first estimation for the length of edges that deliver a good surface approximant. Therefore, all edges $\overline{\mathbf{pq}}$ with empty β-environment and length less than or equal to $r(\mathbf{p})$ can be stored as virtual edges.

In the second step, the radii of points in the neighborhood of each point are used to apply a controlled enlargement of these radii. Therefore, all points $\mathbf{p}, \mathbf{q}$ with intersecting spheres are updated by the maximum of $r(\mathbf{p})$ and $d(\mathbf{p}, \mathbf{q})$ if the β-environment $E_\beta(\mathbf{p}, \mathbf{q})$ of the edge between them is empty of points. Additionally, all points $\mathbf{q}$ that are inside a sphere of a point $\mathbf{p}$ are updated with the maximum of $r(\mathbf{q})$ and their distance to the border of the sphere of $\mathbf{p}$. Since the update of some radii might cause new intersections with other spheres, the second step is repeated until no more update takes place. The generated radii of the second step are used to determine another set of virtual edges that are established between points $\mathbf{p}, \mathbf{q}$ with intersecting spheres and whose connecting edge $\overline{\mathbf{pq}}$ has an empty β-environment. Because all radii have been generated with respect to the distribution of points in the neighborhood of each point, the total length of each virtual edge is limited. Therefore, the virtual edges can be considered as good surface approximants. An analysis that the limitation of the length of graph edges indeed delivers "good surface approximants" is given in the next two chapters.

The third and last step is used to determine the order how the virtual edges of all points are inserted as final edges of the graph. Here, the edges are considered in order of increasing length, since we assume that shorter edges are better surface approximants than longer ones.

The algorithm should work well if at a vertex $\mathbf{p}$ with a bridge edge in its β-environment graph, vertices not adjacent to $\mathbf{p}$ are in a neighborhood of a radius far less than the length of the edge.

Algorithm 5.1 summarizes this procedure in pseudocode notation.

Figures 5.9 and 5.10 show the resulting clustered β-environment graphs of our test cases for $\beta = 1$, $\frac{1}{2}$, and 0. As can be noticed, the number of long edges has significantly reduced. The statistics of vertex degrees of these graphs, that is the number of incident edges to a point, are given in Tables 5.4, 5.5, 5.6. The difference of the number of edges of the clustered β-environment graphs in comparison to the complete β-environment graphs are given in Table 5.7.

5.3 Computational Issues

The computation of the planar relatives of our β-environment graphs, the β-neighborhood graphs, has been intensively investigated in the thesis of Rao [Rao98]. Rao also has presented an approach to the calculation of special three-dimensional cases, namely β-environment graphs for $\beta \geq 1$.

The straightforward approach to the calculation of the β-environment graph is to check all pairs of points for whether they have an empty environment. This approach may be feasible for very small point sets, but cannot be applied to larger sets. A better alternative is the filtering approach. The filtering approach uses a reasonable supergraph G' of the desired graph G, and removes those edges from G' which do not satisfy the empty-environment criterion.

A natural supergraph for $\beta \geq 0$ is the Delaunay triangulation. The Delaunay triangulation often works quite well, but it may have $\mathcal{O}(n^2)$ edges in 3D space in the worst case, n the number of vertices.

An idea of constructing an other type of supergraph, the *sector supergraph*, can be found in the work of Yao [Yao82, Rao98]. For every point $\mathbf{p}$, the space is subdivided into pyramidal sectors R with apex $\mathbf{p}$. If the sectors are sufficiently narrow, it can be shown that every point in R which is more distant than a certain bound dependent on the distance of the nearest neighbor of $\mathbf{p}$ in the sector does not have an empty environment. The supergraph G' contains all connections from $\mathbf{p}$ to those points in R as edges which do not satisfy this criterion. For $\beta \geq 1$, Yao has shown that the sector supergraph only contains $\mathcal{O}(n)$ edges. The edges of the supergraph can be determined by nearest-neighbor search as described in Appendix B.

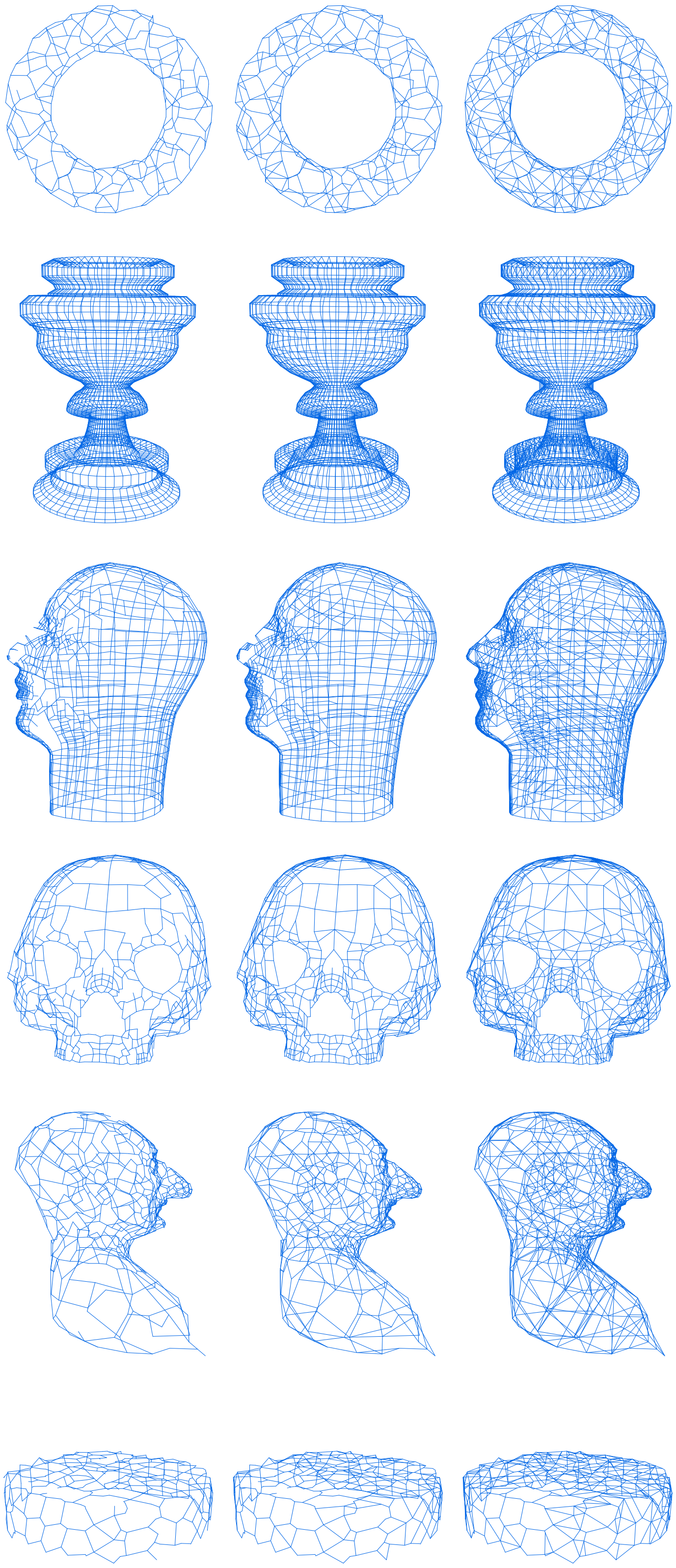


Figure 5.9: The clustered β-environment graphs for $\beta = 1$, $\frac{1}{2}$, and 0.

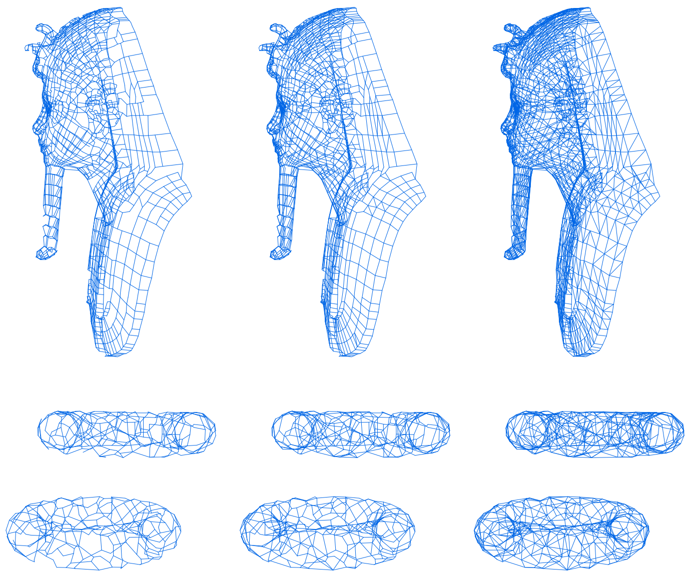


Figure 5.10: The clustered β-environment graphs for $\beta = 1, \frac{1}{2}, 0$.

	# of vertices of degree k in the clustered 1-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$k=1$	5	0	20	21	14	13	14	13
$k=2$	105	0	271	193	180	158	447	211
$k=3$	175	100	474	349	396	179	1104	343
$k=4$	25	2550	722	133	105	21	721	53
$k=5$	0	0	0	2	0	0	0	0
# edges	420	5250	2436	998	991	475	3552	838

Table 5.4: The statistics of vertex degrees for the clustered 1-environment graph.

Independent from the type of supergraph, the main remaining algorithmic task is to check the emptiness of the environments of the edges efficiently. An approach is to consider the set of all environments together, and check the containment of points by e.g. space-sweeping. Another alternative is to preprocess the points for efficient query processing with the environment of the edges. Yao has proposed solutions for that mainly for the planar case, under the view of worst-case efficient data structures.

As a heuristic approach we propose to proceed like for the k-nearest neighbor search of Appendix B. For the Delaunay triangulation as supergraph, the Delaunay triangulation can be used for the search procedure, too.

For the sector graph, a hierarchical tetrahedrization defined as follows is useful. A hierarchical tetrahedrization is given by a tree. The nodes of the tree represent tetrahedra. The root tetrahedron envelops all other tetrahedra. Every non-leaf-tetrahedron t has succeeding tetrahedra which are defined by a split point **p** that splits the tetrahedron into four sub-tetrahedra. The tetrahedrization defined by a hierarchical tetrahedrization consists of the tetrahedra at its leaves. The advantage of this tetrahedrization is that it can easily be manipulated.

In order to check for emptiness, we first search for the tetrahedra incident to the edge e under consid-

	# of vertices of degree k in the clustered $\frac{1}{2}$-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$k = 1$	0	0	4	2	1	3	1	2
$k = 2$	47	0	112	52	53	84	147	97
$k = 3$	159	100	384	318	271	181	774	319
$k = 4$	91	2450	914	295	293	91	1077	168
$k = 5$	13	100	66	30	74	12	261	33
$k = 6$	1	0	7	1	3	0	26	1
# edges	500	5300	2704	1198	1240	569	4193	998

Table 5.5: The statistic of vertex degrees for the clustered $\frac{1}{2}$-EG.

	# of vertices of degree k in the clustered 0-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$k = 1$	0	0	0	0	0	0	0	0
$k = 2$	7	0	18	14	1	31	11	14
$k = 3$	59	51	113	84	31	101	174	124
$k = 4$	121	1626	367	271	227	129	571	237
$k = 5$	82	717	550	217	254	86	736	160
$k = 6$	33	203	414	97	143	17	682	68
$k = 7$	7	52	23	13	28	7	95	16
$k = 8$	1	1	2	2	10	0	13	1
$k = 9$	0	0	0	0	1	0	3	0
$k = 10$	0	0	0	0	0	0	0	0
$k = 11$	0	0	0	0	0	0	0	0
$k = 12$	0	0	0	0	0	0	1	0
# edges	670	5916	3627	1569	1708	731	5704	1338

Table 5.6: The statistic of vertex degrees for the clustered 0-EG.

eration. Then further tetrahedra are determined successively by using the family of β-environments to define a distance function around e. The vertices of the tetrahedra are checked for containment in the environment $E(e)$. Search is terminated if a vertex is in the environment, or if the distance defined by $E(e)$ is completely exhausted.

Tetrahedrizations have also been used as auxiliary data structure for calculation of the candidate edges of the clustered environment graph.

	# of difference edges between standard and clustered β-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$\beta = 1$	6	50	17	6	8	9	11	30
$\beta = \frac{1}{2}$	4	50	4	1	2	5	10	39
$\beta = 0$	2	226	10	16	7	3	55	69

Table 5.7: The difference of the number of edges of the standard β-EG and the clustered β-EG.

5.4 Discussion

In this chapter more comprehensive classes of graphs which are suitable as skeleton of a surface have been defined and investigated. The problem of bridge-edges has been treated by introducing so-called clustered environment graphs. Chapter 7 will present arguments that this approach is indeed successful.

Chapter 6

Approximation and Reconstruction

In this chapter, we fix the notion of a good reconstruction. We do that by considering the embeddability of a constructed mesh onto the surface from which the finite input point set has been sampled. We favorize a special type of embedding, called *nearest–neighbor–embedding*. We show that meshes with sufficiently short edges and triangles of not too large angles can be embedded in that sense. Because the existence of a mesh which can be embedded is necessary in order that a reconstruction algorithm can find an embeddable mesh at all, these investigations have implications on the choice of the sampling sets to which a reconstruction algorithm is applied: the algorithm has to be able to construct meshes which satisfy the recognized conditions for the existence of a nearest-neighbor embedding.

6.1 Surface Approximation and Reconstruction

The following definition fixes the notion of approximation of a surface by a straight-line manifold 2D-CC and the notion of surface reconstruction.

Definition 6.1 (Reconstruction) *Let us consider a given surface S and a straight-line manifold 2D-CC M with vertices on S. A geometric manifold 2D-CC M' over the vertices of M which are on S is called an **embedding** of M into S if there is a continuous function from M to S which*

(1) is a one-to-one mapping between M and M',

(2) is the identity on the vertices,

(3) maps edges to edges, and faces to faces.

*M is called **embeddable** into S if an embedding M' of M into S exists. A surface is called **reconstructible** from a finite set P of sampling points on S if a straight-line manifold 2D-CC M with vertex set P exists so that M is embeddable into S.*

An example of an embedding is the nearest-neighbor embedding.

Definition 6.2 (Nearest-neighbor (NN) embedding) *Let S, M, and M' be defined like in the preceding definition. The **nearest-neighbor (NN) image** of M on S is the set of all points $\mathbf{p}' \in S$ for which a point $\mathbf{p} \in M$ exists so that $\mathbf{p}'$ is a nearest neighbor of $\mathbf{p}$ on S.*
*M' is called a **nearest-neighbor (NN) embedding** of M if*

(1) every point on M has a unique nearest-neighbor-point on the surface S,

(2) M' is an embedding of M under the nearest-neighbor (NN) mapping.

Condition (1) is required because in general a point in space may have more than one nearest neighbor, and thus the mapping of M to its nearest neighbor image needs not to be a function in the sense necessary for the definition of ”embedding”.

Not all surfaces are suitable for NN-embeddings. For example, if a surface has a sharp edge, like for example a cube, points in space arbitrarily close to the surface exist which do not have a unique nearest neighbor. The type of surface which is subject of the following definition is favorable for the concept of NN-embedding.

Definition 6.3 (Save-fringe (SF) surface) *Let be $r > 0$, and S be a closed surface which possesses a tangent plane at every point. At $\mathbf{p} \in S$ we consider the two closed balls of radius r tangent to S at $\mathbf{p}$. If $\mathbf{p}$ is the only common point of those two balls with the surface, $\mathbf{p}$ is called r-**save**. If all points $\mathbf{p} \in S$ are r-save, the surface is called a **save-fringe (SF) surface**. r is called a **save-fringe (SF) radius** of S.*

*The set of all points of a shortest distance to S less than or equal to $r > 0$ is called the r-**fringe** of S. If r is a save-fringe radius, then the fringe is called **save**.*

The concept of SF-surfaces can be extended to surfaces with boundary, too, but we do not include this case in order to keep the presentation simple. Furthermore, the constant SF-radius can be replaced with a function $r : S \to \mathbb{R}_+$. Then a point $\mathbf{p} \in S$ is called $r(\mathbf{p})$-save if the two closed tangent balls at $\mathbf{p}$ of radius $r(\mathbf{p})$ do just have $\mathbf{p}$ in their intersections with S. In order to guarantee saveness, the function $r(\mathbf{p})$ has to be chosen so that the balls do not reach the medial axis of S. The medial axis of a surface is defined as the set of all points in space which have at least two closest points on the surface. By using a function instead of a constant r, adaptivity to the surface behavior can be achieved. In order to simplify the presentation, we will however restrict ourselves to the non-adaptive case.

Theorem 6.4 (Properties of SF-fringes) *Let S be a compact SF-surface without boundary and with SF-radius $r > 0$.*

(1) Every point of the r-fringe of S has a unique nearest neighbor on S.

(2) If the length of a line segment $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in S$, is bounded by r, s is a subset of the r-fringe.

(3) If the edge length of a triangle $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$, $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$, is bounded by r, t is a subset of the r-fringe.

Proof: Let $\mathbf{q}$ be a point in space with distance less than r from S. We consider the ball B with center $\mathbf{q}$ through a nearest neighbor $\mathbf{q}'$ of $\mathbf{q}$ on S. B is tangent to S at $\mathbf{q}'$. Let B_r be the ball tangent at $\mathbf{q}'$ of radius r located on the same side as B. Then B is a subset of B_r and thus does not contain any further points of S. Hence $\mathbf{q}'$ is unique, and thus (1) holds.

If the length of s or the edge length of t, respectively, is less than r, and $\mathbf{p}$ is a vertex of s respectively t, then s and t are completely in the open ball $B_r(\mathbf{p})$ of radius r and with center $\mathbf{p}$. For every point $\mathbf{p} \in S$, the open ball $B_r(\mathbf{p})$ is a subset of the r-fringe because all its points have distance less than r from S. Thus s and t are subsets of the r-fringe, that is (2) and (3) hold. ∎

6.2 Sufficient Conditions for NN-Embeddable Line Segments

The following theorem gives sufficient conditions for the existence of NN-embeddings of single line segments for SF-surfaces.

Theorem 6.5 (Sufficient condition for NN-embeddable line segments) *Let S be a compact SF-surface without boundary and with SF-radius r. If a line segment s or a triangle t is completely in the r-fringe, then the NN-function* **f** *(which exists by (1) of the preceeding theorem) from s and t, respectively, is continuous. In the case of s,* **f** *additionally is one-to-one. Thus, in that case, the image under* **f** *is an embedding of s.*

Proof: In the following we treat s and t simultaneously, and call them "object". The proof of continuity of **f** is by contradiction. We assume that there is a point **u** in our object at which **f** is not continuous. That means that a sequence $\mathbf{u}_i$, $i = 1, \ldots, \infty$, of points in **u** with $\lim_{i\to\infty} \mathbf{u}_i = \mathbf{u}$ exists for which $||\mathbf{f}(\mathbf{u}_i) - \mathbf{f}(\mathbf{u})|| \geq \varepsilon_0$ for some $\varepsilon_0 > 0$, $i = 1 \ldots, \infty$. Because S is assumed to be compact, $\mathbf{f}(\mathbf{u}_i)$ has an accumulation point $\mathbf{f}^*$ with $||\mathbf{f}^* - \mathbf{f}(\mathbf{u})|| \geq \varepsilon_0$. That means that a subsequence $\mathbf{u}_{i_k}$, $k = 1, \ldots, \infty$, exists with $\lim_{k\to\infty} \mathbf{f}(\mathbf{u}_{i_k}) = \mathbf{f}^*$.

Claim: $||\mathbf{u} - \mathbf{f}(\mathbf{u})|| = ||\mathbf{u} - \mathbf{f}^*||$.

If the claim would be wrong, then $||\mathbf{u} - \mathbf{f}(\mathbf{u})|| < ||\mathbf{u} - \mathbf{f}^*||$, because $\mathbf{f}(\mathbf{u})$ is the closest surface point. Let $\varepsilon_1 := |d(\mathbf{u}, \mathbf{f}^*) - d(\mathbf{u}, \mathbf{f}(\mathbf{u}))|/2$. Then $r_1 > 0$ exists so that for all points $\mathbf{u}'$ of our object with $||\mathbf{u}-\mathbf{u}'|| < r_1$, $||\mathbf{u}'-\mathbf{f}(\mathbf{u})|| < ||\mathbf{u}-\mathbf{f}(\mathbf{u})||+\varepsilon_1$. Because k_0 exists so that for all $k > k_0$, $||\mathbf{u}-\mathbf{u}_{i_k}|| < r_1$, we have $||\mathbf{u} - \mathbf{f}^*|| = \lim_{k\to\infty} ||\mathbf{u}_{i_k} - \mathbf{f}(\mathbf{u}_{i_k})|| \leq \lim_{k\to\infty} ||\mathbf{u}_{i_k} - \mathbf{f}(\mathbf{u})|| \leq ||\mathbf{u} - \mathbf{f}(\mathbf{u})|| + \varepsilon_1 = (||\mathbf{u} - \mathbf{f}^*|| + ||\mathbf{u} - \mathbf{f}(\mathbf{u})||)/2 < ||\mathbf{u} - \mathbf{f}^*||$, what cannot hold. This proves the claim.

We now have two closest surface points $\mathbf{f}(\mathbf{u})$, $\mathbf{f}^*$ of **u**, and hence a contradiction. This proves continuity of **f**.

If **f** would not be one-to-one on a line segment $s = \overline{\mathbf{pq}}$, then two points **r** and **s** on s would exist which map to the same closest point $\mathbf{p}' \in S$. $\mathbf{p}'$ may be equal to **p** or to **q**, or an intersection point of s and S. **r** and **s** are on a common line which traverses $\mathbf{p}'$ in direction of its normal. But this means that s is part of that line. Let $\mathbf{q}'$ be a point of S so that $s' := \overline{\mathbf{p}'\mathbf{q}'}$ is a sub-segment of s not intersecting S, cf. Figure 6.1.

Let B_r be the ball tangent to S at $\mathbf{p}'$ of radius r with center on the line induced by s' and on the same side of S like s'. Because r is an SF-radius, B_r is free of points of S, and thus s' completely traverses B_r. This means in particular that the center $\mathbf{r}_0$ of B_r is on s'. Because of Theorem 6.4 (1), $\mathbf{p}'$ is the unique nearest neighbor $\mathbf{f}(\mathbf{r}_0)$ on S.

Let F be the plane perpendicular to s' through $\mathbf{r}_0$, $H(\mathbf{p}')$ the open half-space induced by F which contains $\mathbf{p}'$, and $\overline{H}(\mathbf{p}')$ the opposite open half-space, cf. Figure 6.1. Because **f** is continuous by the first part of this theorem, an open environment $e(\mathbf{r}_0) \subseteq s'$ of $\mathbf{r}_0$ exists so that $\mathbf{f}(e(\mathbf{r}_0)) \subseteq H(\mathbf{p}') \cap S$.

On the other hand, because s and thus s' is assumed to be in the r-fringe of S, the distance of all points of s' to S is less than or equal to r. Thus the nearest neighbors of all points $\mathbf{r}' \in s' \cap \overline{H}(\mathbf{p}')$ are located in $\overline{H}(\mathbf{p}')$. The reason is that their nearest neighbors are in the difference of the ball of radius r with center $\mathbf{r}'$ minus B_r, which is a subset of $\overline{H}(\mathbf{p}')$. But this means that $\mathbf{f}(e(\mathbf{r}_0))$ contains points in $\overline{H}(\mathbf{p}')$ which is a contradiction to the result of the end of the preceding paragraph. ∎

6.3 Sufficient Conditions for NN-Embeddable Triangles

In order to give sufficient conditions for the existence of NN-embeddings for triangles, we need a different view on surfaces:

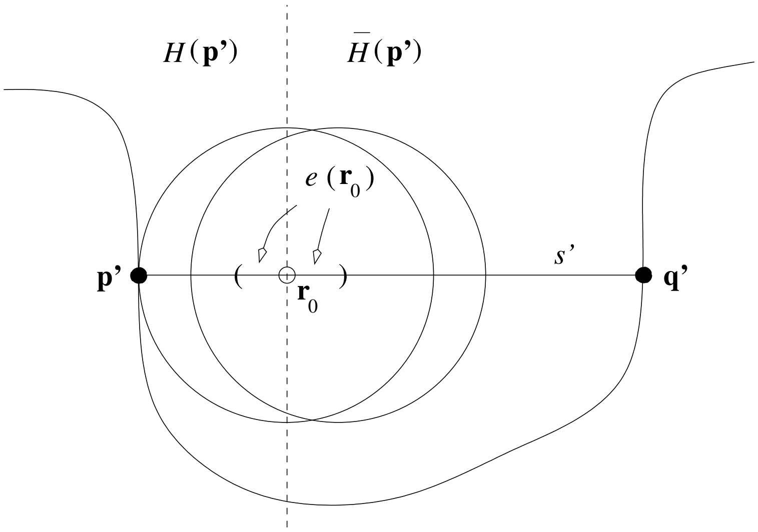


Figure 6.1: Illustration of the proof of Theorem 6.5.

Definition 6.6 (Bounded curvature condition) *Let S be a compact surface without boundary and with a unique normal direction at every point. For $\delta > 0$, $\mathbf{p} \in S$, let $E_\delta(\mathbf{p})$ be the connected component of the points $\mathbf{q} \in S$ with $d(\mathbf{p}, \mathbf{q}) < \delta$ which contains $\mathbf{p}$. S satisfies the **bounded curvature condition** if for every $\alpha > 0°$ a $\delta > 0$ exists so that for all points $\mathbf{p}$ in S and all points $\mathbf{q} \in E_\delta(\mathbf{p})$ the absolute angle between the surface normals $\mathbf{n}(\mathbf{p})$ and $\mathbf{n}(\mathbf{q})$ is less than α.*

The following theorem shows that SF-surfaces have the bounded curvature condition.

Theorem 6.7 (Bounded curvature condition of SF-surfaces) *Compact SF-surfaces without boundary satisfy the bounded curvature condition.*

Proof: The theorem is proved by contradiction. We assume that there is an SF-surface S and an SF-radius r_0 for which a point $\mathbf{p} \in S$, an $\alpha_0 > 0°$, and a sequence of points $\mathbf{p}_i$, $i = 1, \ldots, \infty$ with $\lim_{i\to\infty} \mathbf{p}_i = \mathbf{p}$ exist for which the angles α_i between the normals of $\mathbf{p}$ and $\mathbf{p}_i$ satisfy $\alpha_i \geq \alpha_0$.
We consider points $\mathbf{p}_i$ which are closer to $\mathbf{p}$ than those points on the surface of the two balls whose normals have an angular deviation of $\alpha_0/2$ from the normal $\mathbf{n}$ of $\mathbf{p}$. Let $\mathbf{n}_i$ denote the normal of $\mathbf{p}_i$. We consider the intersection of the configuration with the unique plane E_i which contains $\mathbf{p}_i$, $\mathbf{p}$, and the vector $\mathbf{n}_i$ at $\mathbf{p}_i$. The trace of the two balls in E_i are two discs (Figure 6.2). They induce two wedges in one of which $\mathbf{p}_i$ is located. Let l_i be the common tangent of the discs at $\mathbf{p}$ in plane E_i. Two cases for $\mathbf{n}_i$ can be distinguished: the projection of $\mathbf{n}_i$ on l_i either has a positive component in direction of $\mathbf{p}$, or a negative component. In the positive case, we move $\mathbf{p}_i$ and its two tangent balls in direction of $\mathbf{n}_i$ onto the boundary of the closest ball. In the negative case, $\mathbf{p}_i$ and its balls is moved into the opposite direction of $\mathbf{n}_i$. Let $\mathbf{p}'_i$ be the resulting location of $\mathbf{p}_i$ on the boundary.
In the positive case we consider that one of the two balls at $\mathbf{p}'_i$ for which $\mathbf{n}_i$ shows into its interior, while for the negative case we take the ball for which $\mathbf{n}_i$ is an outer normal. If $\mathbf{p}_i$ is sufficiently close to $\mathbf{p}$, the disc induced by intersecting the ball with E_i contains $\mathbf{p}$, and thus the ball contains $\mathbf{p}$. Because of $\lim_{i\to\infty} \mathbf{p}_i = \mathbf{p}$, a suitable $\mathbf{p}_i$ exists. Furthermore, it can be observed that $\mathbf{p}$ stays in the disc, and thus in the ball, if $\mathbf{p}'_i$ and its balls is moved back straightline to $\mathbf{p}_i$. Thus one of the balls at $\mathbf{p}_i$ contains the surface point $\mathbf{p}$. That contradicts to the SF-property of the surface. ∎

A further ingredient in order to reach our goal of formulating sufficient conditions for the existence of NN-embeddings of triangles is the behaviour of normals. The following theorems make assertions on

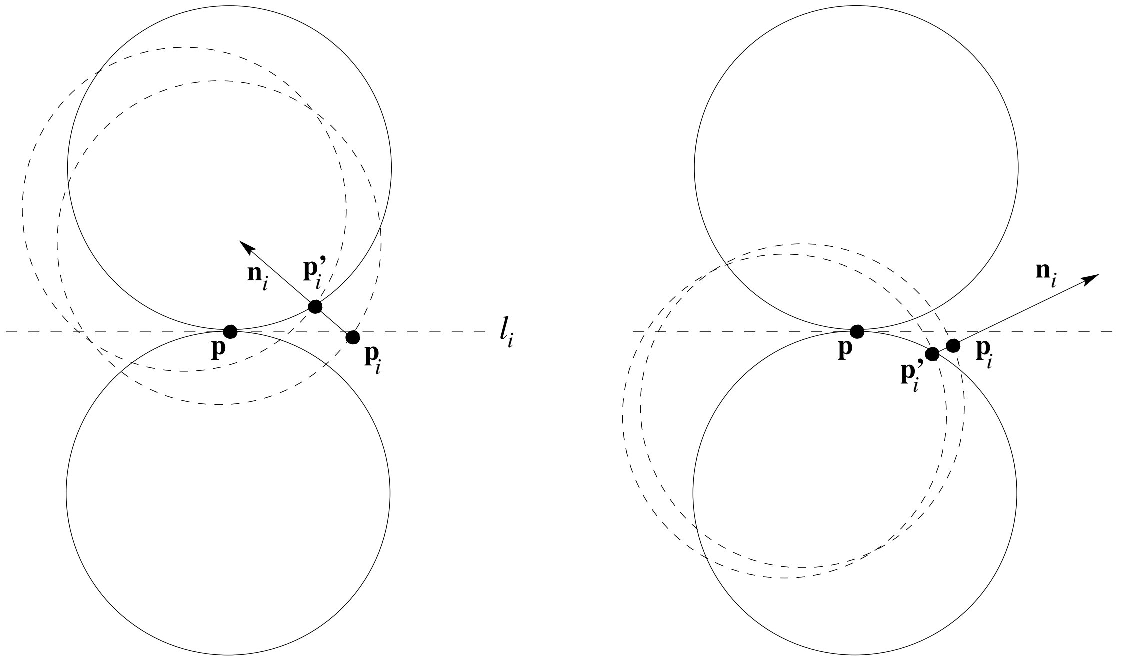


Figure 6.2: Illustration for the proof of the bounded curvature condition for SF-surfaces.

that.

Theorem 6.8 (Deviation of NN-embedded triangle normals) *Let S be a compact SF-surface without boundary, and $\alpha_0 > 0°$. Then a save-fringe radius $r_0 > 0$ exists so that for all triangles $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$, $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$, with edge length at most r_0, the deviation of the normals at the points of the NN-embedding $\mathbf{f}(t)$ is less than α_0.*

Proof: Let $\delta_0 > 0$ and $r_0 > 0$ be chosen so that

- δ_0 satisfies the condition of the bounded curvature condition of Definition 6.6 for α_0, and
- r_0 is an SF-radius of S so that $\delta_0 = 2r_0$.

Let $h(t)$ be the maximum distance of points of a triangle t from S. Because $h(t) \leq r_0$, $r_0 + h(t) \leq r_0 + r_0 = \delta_0$. Since the edge length of t is bounded by r_0, the distance of $\mathbf{q}$ to any point $\mathbf{f}(\mathbf{q}')$, $\mathbf{q}' \in t$, of the NN-embedding $\mathbf{f}(t)$ satisfies $||\mathbf{q} - \mathbf{f}(\mathbf{q}')|| \leq ||\mathbf{q} - \mathbf{q}'|| + h(t) \leq r_0 + h(t) \leq \delta_0$. Thus $\mathbf{f}(t)$ is completely located in the ball of radius δ_0 and center $\mathbf{q}$. Because S is an SF-surface, all points of $\mathbf{f}(t)$ have unique normals. Since the NN-embedding $\mathbf{f}$ is continuous on t, the image $\mathbf{f}(t)$ is connected and, because $\mathbf{q} = \mathbf{f}(\mathbf{q}) \in \mathbf{f}(t)$, $\mathbf{f}(t)$ is a subset of $E_{\delta_0}(\mathbf{q})$. Thus, by the bounded curvature condition, the deviation of the normals at the points of the NN-embedding $\mathbf{f}(t)$ is less than α_0. ∎

Theorem 6.9 (Deviation from vertex normal) *Let S be a compact SF-surface without boundary and r_0 be an SF-radius of S. Let $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ be a triangle and $0° < \gamma_{\min} < 180°$, $\alpha_0 > 0°$ so that*

(1) the length of the edges $\overline{\mathbf{pq}}$ and $\overline{\mathbf{rq}}$ is less than $l_0 := 2r_0 \sin(\alpha_0) \sin(\frac{\gamma_{\min}}{2})$,

(2) the angle γ at $\mathbf{q}$ exceeds $\gamma_{\min}$.

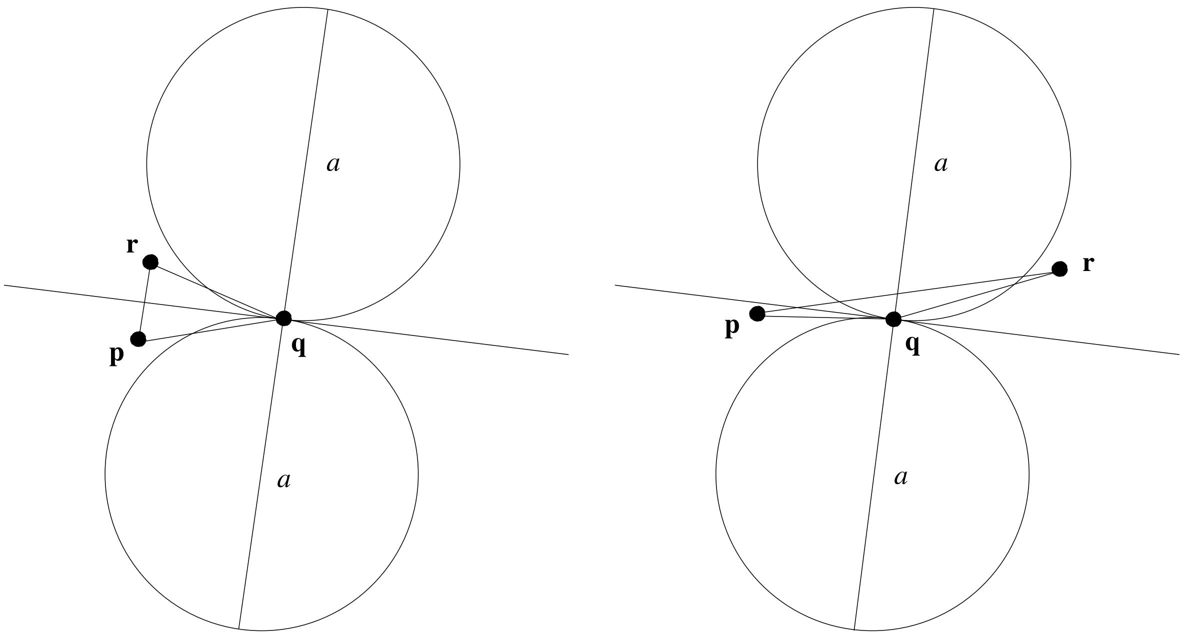


Figure 6.3: The two cases of the location of a triangle $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ with respect to the tangent balls at a surface S, depicted in the slicing plane spanned by the triangle $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$.

If the angle α between the surface normal at $\mathbf{q}$ *and the normal of the triangle exceeds α_0, then the angle γ exceeds $\gamma_{\max} := 180^\circ - \gamma_{\min}$. Equivalently, if the angle γ does not exceed $\gamma_{\max}$, then the angle between the surface normal at* $\mathbf{q}$ *and the normal of the triangle does not exceed α_0.*

Proof: We consider a triangle $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$, and the two tangential balls located at $\mathbf{q}$.
Let H be the plane induced by $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$. The intersection of the configuration with H consists of the triangle $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ and two discs induced by the balls. The tangent plane at $\mathbf{q}$ induces a tangent line at $\mathbf{q}$ of the two discs. $\mathbf{p}$ and $\mathbf{r}$ are outside the discs. Figure 6.3 shows the configuration on H.

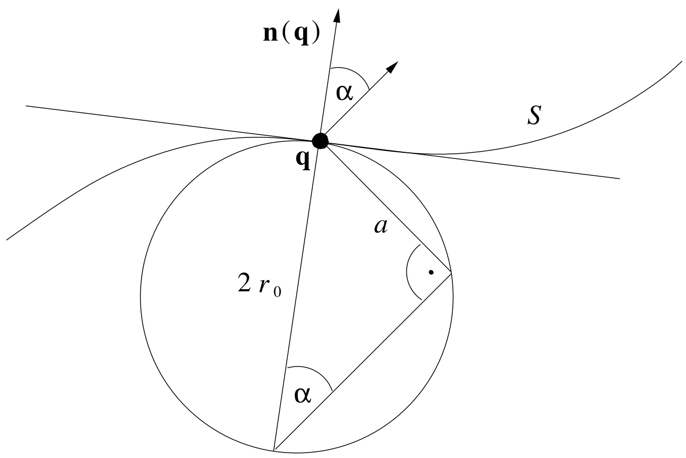


Figure 6.4: Slice through the center of a ball along the plane spanned by the surface normal $\mathbf{n}(\mathbf{q})$ and the normal of a plane.

The intersection of a plane for which the normal deviates from that of the surface by an angle α is a disc of diameter

$$a = 2r_0 \sin(\alpha),$$

cf. Figure 6.4.

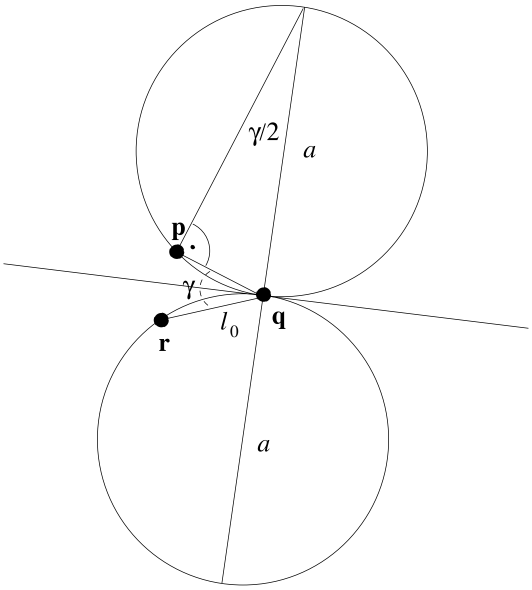


Figure 6.5: Estimation of the maximum angle at a triangle of edge length $\leq l_0$ whose vertices are on the same side of the pair of discs.

Because $\gamma > \gamma_{\min}$ by (2), and

$$l_0 = 2r_0 \sin(\alpha_0) \sin\left(\frac{\gamma_{\min}}{2}\right) < 2r_0 \sin(\alpha) \sin\left(\frac{\gamma_{\min}}{2}\right)$$

by (1), and the condition that α exceeds α_0, **p** and **r** are located in different sectors of the double wedge induced by the boundary circles of the two balls at **q**. Figure 6.5 illustrates the impossibility of

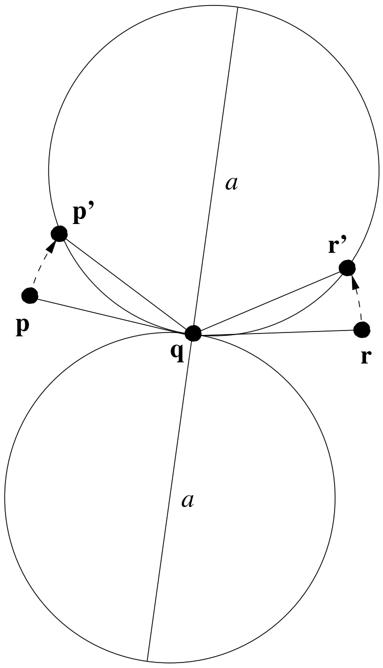


Figure 6.6: Mapping of the points **p** and **r** onto the circle by rotation around **q**.

the case that **p** and **q** are in the same sector. We rotate **p** and **r** around **q** so that they are located on the boundary of one of the discs. The rotation is chosen so that the new sector is a subsector of the sector in which the triangle lies, cf. Figure 6.6. The angle at **q** of the new triangle does not exceed that of the original one. We will show that if we use the new angle instead of the old one, the theorem is satisfied.

On the boundary of that disc, we consider three points $\mathbf{p}', \mathbf{q}', \mathbf{r}'$ with $d(\mathbf{p}', \mathbf{q}') = l, d(\mathbf{q}', \mathbf{r}') = l$. The angle γ at $\mathbf{q}'$ satisfies

$$\sin\left(\frac{\gamma}{2}\right) = \sqrt{1 - \left(\frac{l}{a}\right)^2},$$

cf. Figure 6.7.

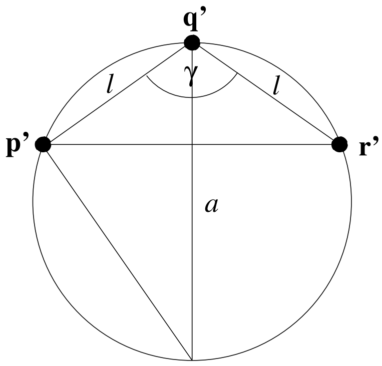


Figure 6.7: Calculation of the angle γ from l and a.

Replacing a in that formula with the right hand side of the previous one we get

$$\sin\left(\frac{\gamma}{2}\right) = \sqrt{1 - \left(\frac{l}{2r_0 \sin(\alpha)}\right)^2}.$$

If we bound the edge length by l_0 and the angle α by α_0, we get

$$\sin\left(\frac{\gamma}{2}\right) \geq \sqrt{1 - \left(\frac{l_0}{2r_0 \sin(\alpha_0)}\right)^2}.$$

By replacing l_0 by its definition in (3), we get

$$\sin\left(\frac{\gamma}{2}\right) \geq \cos\left(\frac{\gamma_{\min}}{2}\right) = \sin\left(\frac{\gamma_{\max}}{2}\right),$$

and hence $\gamma \geq \gamma_{\max}$.

If the edge lengths are less than l, the angle γ increases. That implies that the angle bound holds for all triangles with that bound on the edge length.

Because the size of the angle γ of the original triangle of the surface is at least that of the modified one, the bound holds for the triangles on the surface. This proves the assertion of the theorem. ∎

The theorem tells us that if an angle of the triangles of a triangulation is bounded by lower and upper constant bounds, the deviation of the triangle's normal from the surface normals at the vertices of the triangle is less than an arbitrary small given constant angle bound if the lengths of the edges are sufficiently small.

Lemma 6.10 *Let S be a compact SF-surface without boundary, t a triangle with vertices on S, and the angle between $\mathbf{n}(t)$ and any normal on the nearest-neighbor image $\mathbf{f}(t)$ of t be less than $\alpha_0 < 90°$. Then the NN-image $\mathbf{f}(t)$ of t is an NN-embedding.*

Proof: The condition of the lemma implies that none of the normals in $\mathbf{f}(t)$ is co-linear to a vector $\mathbf{q}' - \mathbf{p}'$ for $\mathbf{p}', \mathbf{q}' \in t$, $\mathbf{p}' \neq \mathbf{q}'$. But that means that $\mathbf{q}'$ and $\mathbf{p}'$ cannot map to a common nearest neighbor on S. Thus the NN-image $\mathbf{f}(t)$ of t is an NN-embedding. ∎

Lemma 6.11 *Let S be a compact SF-surface without boundary, $0° < \gamma_{\min} < \gamma_{\max} := 180° - \gamma_{\min} < 180°$, $0° < \alpha_0 < 90°$. Then an SF-radius $r_0 > 0$ of S exists so that for all triangles $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$, $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$, for which*

(1) the edge length of t is bounded by r_0,

(2) the angle γ at $\mathbf{q}$ satisfies $\gamma_{\min} < \gamma < \gamma_{\max}$,

the angle between the normal $\mathbf{n}(t)$ of the triangle t and any normal on $\mathbf{f}(t)$ is less than $\alpha_0 < 90°$.

Proof: From Theorem 6.8 we know that for a given $\alpha_1 > 0$, an SF-radius r_0 of S exists so that for all triangles $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$, $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$, satisfying (1) the deviation of the normals at the points of the NN-embedding $\mathbf{f}(t)$ is less than α_1. We set $r_1 := r_0$, r_0 the SF-radius of Theorem 6.8.
From Theorem 6.9 we know that for a given α_2, an $l_0 > 0$ exists so that, if (1) and (2) are satisfied, the angle between the surface normal $\mathbf{n}(\mathbf{q})$ at $\mathbf{q}$ and the normal $\mathbf{n}(t)$ of the triangle does not exceed α_2. We set $r_2 := l_0$, l_0 as in Theorem 6.9.
We now define $\alpha_1 = \alpha_2 := \alpha_0/2$, $r_0 := \min\{r_1, r_2\}$. Then the angle between $\mathbf{n}(t)$ and any normal on $\mathbf{f}(t)$ is less than $\alpha_0 < 90°$. That is the assertion of the Lemma. ∎

With help of these results, we now can give conditions which are sufficient for the existence of NN-embeddings of triangles.

Theorem 6.12 (Sufficient condition for NN-embeddable triangles) *Let S be a compact SF-surface without boundary, $0° < \gamma_{\min} < \gamma_{\max} := 180° - \gamma_{\min} < 180°$. Then an SF-radius $r_0 > 0$ of S exists so that for all triangles $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$, $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$, for which*

(1) the edge length of t is bounded by r_0,

(2) the angle γ at $\mathbf{q}$ satisfies $\gamma_{\min} < \gamma < \gamma_{\max}$,

the NN-image $\mathbf{f}(t)$ of t is an NN-embedding.

Proof: The conditions of the theorem imply by Lemma 6.11 that the angle between the normal $\mathbf{n}(t)$ of the triangle t and any normal on $\mathbf{f}(t)$ is less than $\alpha_0 < 90°$. By Lemma 6.10 this assertion implies that the NN-image $\mathbf{f}(t)$ of t is an NN-embedding. ∎

6.4 Sufficient Conditions for NN-Embeddable Pairs of Triangles

The next step is to show that more than one triangle can also be embedded without intersections if some additional constraints hold. These constraints concern the angle between triangles, the so-called dihedral angle, which has been introduced in Definition 5.13 of Chapter 5.

Corollary 6.13 *Let S be a compact SF-surface without boundary. Let $0° < \gamma_{\min} < \gamma_{\max} := 180° - \gamma_{\min} < 180°$, $0° < \alpha_0 < 90°$, and r_0 as in Lemma 6.11. Let t_1 and t_2 be two triangles which*

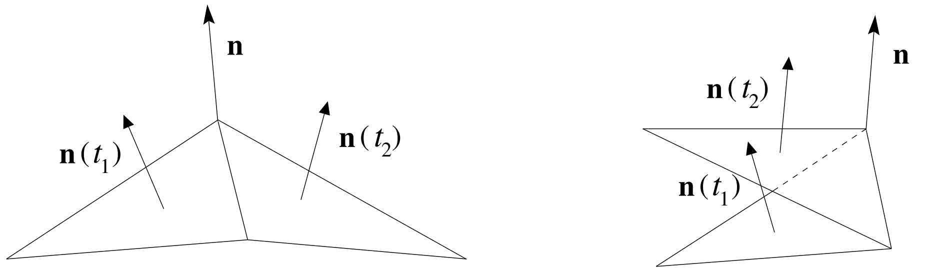


Figure 6.8: Cases of the dihedral angle between two triangles.

(1) have edge lengths less than r_0,

(2) have angles in the interval $(\gamma_{\min}, \gamma_{\max})$,

(3) share a common edge.

Then the dihedral angle between t_1 *and* t_2 *is in the interval* $(180° - 2\alpha_0, 180°]$ *or in the interval* $[0°, 2\alpha_0)$.

Proof: Two triangles which share a common edge also have a common vertex **p**. By Lemma 6.11, the angles $\alpha(\mathbf{n}(\mathbf{p}), \mathbf{n}(t_i))$, $i = 1, 2$, between the surface normal $\mathbf{n}(\mathbf{p})$ at **p** and the normals $\mathbf{n}(t_i)$, $i = 1, 2$, of the triangles are less than α_0. By considering the metric of shortest distances on the unit sphere, that implies that the angle $|\alpha(\mathbf{n}(t_1), \mathbf{n}(t_2))|$ between the normals of the triangles satisfies $|\alpha(\mathbf{n}(t_1), \mathbf{n}(t_2))| \leq |\alpha(\mathbf{n}(\mathbf{p}), \mathbf{n}(t_1))| + |\alpha(\mathbf{n}(\mathbf{p}), \mathbf{n}(t_2))| < 2\alpha_0$.

From the two possible orientations of the normals, two cases concerning the dihedral angle arise (cf. Figure 6.8). The bound on the normals implies the two bounds on the dihedral angles of the corollary. ■

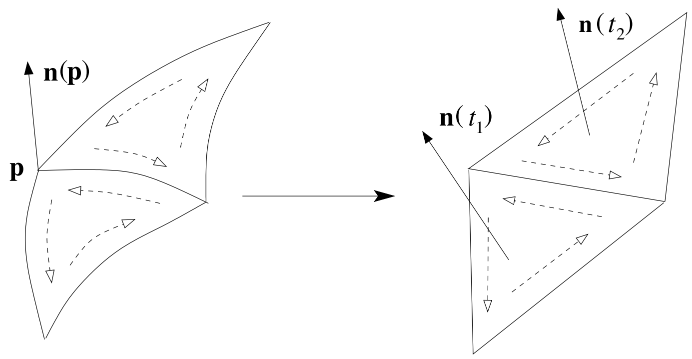


Figure 6.9: Proof of the size of the dihedral angle by transforming the orientation of the surface to the approximating triangle.

For a reasonable triangulation, only the case of dihedral angles in $(180° - 2\alpha_0, 180°]$ is of interest.

Corollary 6.14 (Dihedral angle bound) *Let* S *be a compact SF-surface without boundary. Let* $0° < \gamma_{\min} < \gamma_{\max} := 180° - \gamma_{\min} < 180°$, $0° < \alpha_0 < 90°$, *and* r_0 *as in Theorem 6.12. Let* t_1 *and* t_2 *be two triangles incident to a common edge, which*

(1) have edge length less than r_0,

(2) have angles in the interval $(\gamma_{\min}, \gamma_{\max})$,

(3) are together NN-embeddable in S.

Then the dihedral angle between t_1 *and* t_2 *at the common edge is in the interval* $(180° - 2\alpha_0, 180°]$.

Proof: The NN-embedding of the triangles t_1 and t_2 on S consists of two surface triangles which are disjoint up to parts of their boundaries. We orient the edges of the two surface triangles so that they appear counter-clockwise oriented from the side of the surface on which the normal vector at one of the end points $\mathbf{p}$ of the common edge is directed. We transfer the orientation to t_1 and t_2, and consider the normals $\mathbf{n}(t_1)$ and $\mathbf{n}(t_2)$ directed so that their angles with the normal $\mathbf{n}(\mathbf{p})$ of $\mathbf{p}$ are less than α_0. Then t_1 and t_2 appear counter-clockwise oriented if viewed from the side of their normals. The angle between $\mathbf{n}(t_1)$ and $\mathbf{n}(t_2)$ is less than $2\alpha_0$. Figure 6.9 depicts the resulting configuration. As can be clearly noticed the only possibility of a configuration of that type is that one with the dihedral angle in the interval $(180° - 2\alpha_0, 180°]$. ■

Lemma 6.15 *Let* S *be a compact SF-surface without boundary. Further, let* t_1 *and* t_2 *be two triangles with vertices on* S*, incident to a common edge, with dihedral angle larger than* $180° - \delta_0 > 90°$, $\delta_0 > 0°$, $\mathbf{n}(t_1)$ *and* $\mathbf{n}(t_2)$ *the normals of* t_1 *and* t_2*, respectively,* $\mathbf{p}_1 \in t_1$, $\mathbf{p}_2 \in t_2$. *Then the angle between the vectors* $\mathbf{p}_2 - \mathbf{p}_1$ *and* $\mathbf{n}(t_2)$*, and between* $\mathbf{p}_1 - \mathbf{p}_2$ *and* $\mathbf{n}(t_1)$*, respectively, exceeds* $90° - \delta_0$.

Proof: The smallest angle between $\mathbf{p}_2 - \mathbf{p}_1$ and $\mathbf{n}(t_2)$ is achieved if $\mathbf{p}_2$ is on the common edge of t_1 and t_2, and $\mathbf{p}_2 - \mathbf{p}_1$ perpendicular to that edge. In that case, the angle between $\mathbf{p}_2 - \mathbf{p}_1$ and $\mathbf{n}(t_2)$ is equal to $90° - \delta_0$. The assertion for $\mathbf{p}_1 - \mathbf{p}_2$ and $\mathbf{n}(t_1)$ holds for reasons of symmetry. ■

Now we can formulate a result on intersection-free embeddings of two adjacent triangles.

Theorem 6.16 (Sufficient condition for NN-embeddable pairs of triangles) *Let* S *be a compact SF-surface without boundary,* $0° < \gamma_{\min} < \gamma_{\max} := 180° - \gamma_{\min} < 180°$, $\delta_0 > 0°$. *Then an SF-radius* $r_0 > 0$ *of* S *exists so that all triangles* $t_i = \triangle(\mathbf{p}_i, \mathbf{q}_i, \mathbf{r}_i)$, $\mathbf{p}_i, \mathbf{q}_i, \mathbf{r}_i \in S$, $i = 1, 2$, *for which*

(1) the edge length of t_1 *and* t_2 *is bounded by* r_0,

(2) the angles of t_1 *and* t_2 *at* $\mathbf{q}_1$ *and* $\mathbf{q}_2$*, respectively, satisfy* $\gamma_{\min} < \gamma < \gamma_{\max}$,

(3) the dihedral angle between t_1 *and* t_2 *along a common edge is larger than* $180° - \delta_0 > 90°$,

are NN-embeddable, and their NN-images $\mathbf{f}(t_1)$ *and* $\mathbf{f}(t_2)$ *are disjoint, up to the NN-image of their common edge.*

Proof: From Theorem 6.8 we know that for a given $\alpha_1 > 0°$, $r_0' > 0$ exists so that for all triangles t_1, t_2 satisfying (1) and (2) the deviation of the normals at the points of the NN-images $\mathbf{f}(t_1)$ and $\mathbf{f}(t_2)$ is less than α_1. We set $r_1 := r_0'$.
From Theorem 6.9 we know that for a given $\alpha_2 > 0°$, $r_0' > 0$ exists so that, if (1) and (2) are satisfied, the angle between the surface normal $\mathbf{n}(\mathbf{q}_i)$ at $\mathbf{q}_i$ and the normal $\mathbf{n}(t_i)$ of the triangle t_i, $i = 1, 2$, does not exceed α_2. We set $r_2 := r_0'$.
From Theorem 6.5 we know that an $r_0' > 0$ exists so that all triangles t_1, t_2 satisfying (1) and (2) are NN-embeddable. We set $r_3 := r_0'$.
Let $\alpha_1 = \alpha_2 := (90° - \delta_0)/2$, $r_0 := \min\{r_1, r_2, r_3\}$. For this r_0, the assertion of the theorem that t_1 and t_2 are NN-embeddable holds. Furthermore, the angle between $\mathbf{n}(t_i)$, $i = 1, 2$, and any normal on $\mathbf{f}(t_i)$ is less than $90° - \delta_0 < 90°$. On the other hand, by Lemma 6.15, the directions of lines between points on t_i deviate by more than $90° - \delta_0$ from $\mathbf{n}(t_i)$, $i = 1, 2$. This contradiction implies that none of the lines along the normals on $\mathbf{f}(t_i)$, $i = 1, 2$, intersects t_1 and t_2 simultaneously, except possibly on the common edge of t_1 and t_2. Thus the NN-embeddings are disjoint, up to the common edge. ■

6.5 Discussion

The theorems concerning the NN-embedding state that there is a chance for 2D-CCs with sufficiently short edges that they can be embedded in compact SF-surfaces without boundary. For the choice of the set of sampling points this means that it has to be sufficiently dense.

A reconstruction algorithm should yield a triangular manifold that can be embedded, for example that one that is demanded to exist for the set of sampling points. The consequence of the investigations of this section for a reconstruction algorithm is that it should yield short edges when applied to a dense set of sampling points.

Chapter 7

Analysis of Environment Graphs

As we already know from examples, environment graphs are basically well-suited for reconstruction. In this chapter we present formal arguments for this favorable behavior. We first prove that bridge edges cannot be avoided in general, so that β-EGs without clustering are not useful. Then we show that any given sample set of a compact SF-surface without boundary can be extended so that all edges of the clustered β-EG should have length less than a given bound $l_0 > 0$. Furthermore, we demonstrate that subgraphs of the clustered β-environment graphs are NN-embeddable with high probability if the length of the edges of the subgraph is small, like it should be for the just mentioned clustered β-EG. The chance that such a subgraph is not NN-embeddable can be made arbitrarily small if the maximum edge length decreases to 0. From these observations we then derive the main result that a given sample set of a compact SF-surface without boundary can be extended to a finite point set so that, with high probability, the clustered β-EG, $0 \leq \beta \leq 1$, for this point set should be NN-embeddable into S. Finally, we give examples of sampling strategies which have shown favorable in practical applications.

7.1 Short Edges and non-blockable Line Segments

As we know from Chapter 3, a surface (re-)construction algorithm should yield short edges. In tendency, this requirement is satisfied by the β-environment graphs. The reason is that the probability that the large environment of a long edge contains any other point is high for a dense set of sampling points equally distributed on the surface. However, the examples of Chapter 5 show that long edges occur. An immediate assumption of course is that this effect might be a matter of not sufficiently dense sampling. But as we will show in the following, it may indeed happen that the geometry of the surface forces the occurrence of long edges, although that fortunately is not the standard case.

A set of sampling points is favorable if for any two of its points of larger distance, there is a third point located in the environment of the line segment between the two points.

Definition 7.1 (β-blocked line segment) *Let P be a set of sampling points on a surface S. A line segment $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in S$, is β-**blocked** if a point $\mathbf{r}$ in P exists which is in the open β-environment of s.*

In this definition and in the rest of the chapter we assume $0 \leq \beta \leq 1$, if no other specification is given.

A necessary and sufficient condition that a line segment s can be β-blocked is that the β-environment of s contains a sampling point at all. A line segment with that property is called *blockable*.

Definition 7.2 (β-blockable line segment) *A line segment s between a pair $\mathbf{p}, \mathbf{q}$ of points of a surface is β-**blockable** if the open β-environment of s has a non-empty intersection with the surface.*

The following theorem gives a necessary condition that a line segment is non-blockable.

Theorem 7.3 (Necessary condition for a non-blockable line segment) *Let S be a surface with a unique tangent plane everywhere. For β-environments with $0 \leq \beta \leq 1$, a necessary condition for a non-blockable line segment* $\overline{\mathbf{pq}}$*,* $\mathbf{p}$ *and* $\mathbf{q}$ *inner points of S, is that the line segment is perpendicular to the tangent planes of the surfaces at* $\mathbf{p}$ *and* $\mathbf{q}$.

Proof: If $\overline{\mathbf{pq}}$ is not perpendicular to the tangent plane of the surface at $\mathbf{p}$ (analogously for $\mathbf{q}$), then the plane H through $\mathbf{p}$ perpendicular to $\overline{\mathbf{pq}}$ intersects the surface. There is an inner point $\mathbf{r}$ on S which is on the same side of the plane as the environment $E_\beta(\mathbf{p}, \mathbf{q})$. We consider plane E spanned by $\mathbf{p}, \mathbf{q}, \mathbf{r}$. The curve of intersection of S and E between $\mathbf{r}$ and $\mathbf{p}$ intersects the open environment $E_\beta(\mathbf{p}, \mathbf{q}) \cap E$. Figure 7.1 depicts the configuration for $\beta = 0$. Otherwise that curve would have a tangent at $\mathbf{p}$ in the plane H. The same would hold for a different point $\mathbf{r}'$ whose plane E' does not intersect in a common line. That implies that H is tangent to the surface at $\mathbf{p}$ which contradicts to the assumption made on H. ∎

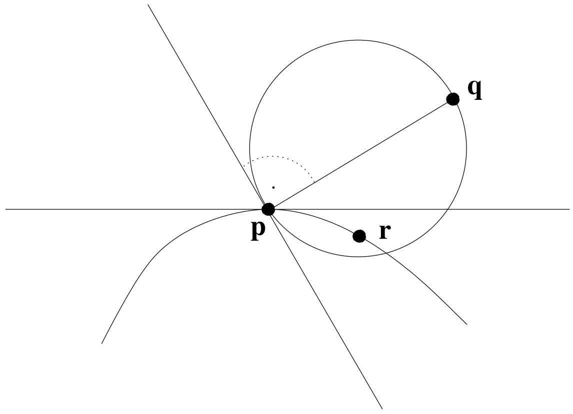


Figure 7.1: The intersection of a surface S with an environment $E_0(\mathbf{p}, \mathbf{q})$, depicted in the slicing plane spanned by $\mathbf{p}, \mathbf{q}$ and $\mathbf{r}$.

Corollary 7.4 *For a surface with unique continuous normals* $\mathbf{n}$ *everywhere, a necessary condition for a line-segment* $\overline{\mathbf{pq}}$ *to be non-blockable is*

$$\mathbf{n}(\mathbf{p})^T(\mathbf{p} - \mathbf{q}) = ||\mathbf{n}(\mathbf{p})|| \cdot ||\mathbf{p} - \mathbf{q}||,$$

$$\mathbf{n}(\mathbf{q})^T(\mathbf{p} - \mathbf{q}) = ||\mathbf{n}(\mathbf{q})|| \cdot ||\mathbf{p} - \mathbf{q}||,$$

where ^{T} denotes the transposition of a vector.

Proof: The first formula expresses that the scalar vector product of the normalized vectors $\mathbf{n}(\mathbf{p})$ and $(\mathbf{p} - \mathbf{q})$ is equal to 1, so that the angle between the two vectors is 0°, the second formula is analogous. That corresponds to the assertion of Theorem 7.3. ∎

The event that $\mathbf{p}$ and $\mathbf{q}$ satisfy both equations of the corollary should in general occur in isolated pairs of points $\mathbf{p}, \mathbf{q}$. In special cases, like for the two parallel tori of Figure 5.10, a curve-like occurrence is possible. For two parallel planes, a surface-like occurrence emerges. Because $\mathbf{n}$ is assumed as a continuous function, the pairs satisfying these equalities define a closed set.

The theorem is formulated for surfaces which have a unique tangent plane everywhere. This constraint in particular excludes boundary points. However an analogous theorem seems also to be possible for boundary curves. We leave the treatment of boundary points open for future work.

The stated condition is not sufficient because the environment may intersect the surface elsewhere.

If we assume the given surface not having non-blockable line segments, the following theorem shows the existence of a sampling set for which the 0-environment graph does not have long edges.

Theorem 7.5 (Existence of length-bounding sampling sets) *Let S be a compact surface without boundary which has just blockable line segments, and P be a finite set of sample points taken from S. Then, for every $l_0 > 0$, P can be extended to a finite point set P' of points of S so that all line segments s with length $l(s) > l_0$ are β-blocked by a point in P', $0 \leq \beta \leq 1$. P' is called a **blocking set**.*

Any further finite extension of P' does not violate this property.

Proof: The proof of existence uses the theorem of finite coverings of compact sets [Ber63, Kur68].

Let M be the set of all line segments between two points of the surface S. M is a compact set. We consider M_1, the set of all line segments of length less than l_0. M_1 is an open set, so that $M_2 := M - M_1$ is a compact set.

For every point $\mathbf{r}$ of S, we consider the set $E(\mathbf{r})$ of all line segments $s = \overline{\mathbf{pq}}$ of M_2 for which $\mathbf{r}$ lies in the interior of the intersection of the surface with the β-environment $E_\beta(\mathbf{p}, \mathbf{q})$ of s. Note that $\mathbf{r} \neq \mathbf{p}$, $\mathbf{r} \neq \mathbf{q}$. $E(\mathbf{r})$ is an open set.

The sets $E(\mathbf{r})$ cover M_2 because every blockable line segment has a surface point in the interior of the intersection of the surface with the diameter sphere.

Because M_2 is compact (it is bounded because S is bounded), there exists a finite covering of M_2 by sets $E(\mathbf{r})$. The points $\mathbf{r}$ of these sets define a blocking set P''.

Because a superset of a blocking set is a blocking set, too, $P' := P \cup P''$ is the desired set of the theorem. ■

For a line segment $s = \overline{\mathbf{pq}}$ with $\mathbf{p}, \mathbf{q} \in P$ which has to be blocked, there is a point $\mathbf{r} \in P$, $\mathbf{r} \neq \mathbf{p}$, $\mathbf{r} \neq \mathbf{q}$, which blocks s. The reason is that, by definition, s is not in $E(\mathbf{p})$ and not in $E(\mathbf{q})$.

Corollary 7.6 *Under the conditions of Theorem 7.5, for every finite super-set P'' of a blocking set P', the edges e of the corresponding β-environment graph on P'' satisfy $l(e) \leq l_0$.*

Definition 7.7 (Samp_1**-property)** *Let be $l_0 > 0$, $\varepsilon > 0$, S a surface. A finite set P of points of S has the $\mathrm{Samp}_1(l_0, \varepsilon)$-property if every line segment $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in S$, has at least one of the following properties:*

(1) $l(s) \leq l_0$,

(2) $\mathbf{n}(\mathbf{p})^T(\mathbf{p} - \mathbf{q}) > ||\mathbf{n}(\mathbf{p})|| \cdot ||\mathbf{p} - \mathbf{q}|| \cdot (1 - \varepsilon)$,
$\mathbf{n}(\mathbf{q})^T(\mathbf{p} - \mathbf{q}) > ||\mathbf{n}(\mathbf{q})|| \cdot ||\mathbf{p} - \mathbf{q}|| \cdot (1 - \varepsilon)$.

That means that s is shorter than l_0, or lies close to line segments which satisfy the necessary condition of non-blockable segments.

Corollary 7.8 (Existence of Samp_1**-sample sets)** *Let S be a compact surface without boundary and with continuous normals, which also may have non-blockable line segments, and P be a finite set of sample points taken from S. For every $l_0 > 0$ and $\varepsilon > 0$, P can be extended to a finite set P' which has the $\mathrm{Samp}_1(l_0, \varepsilon)$-property.*

Proof: Let M_3 be the set of line segments satisfying (2) of Definition 7.7. Then $M_4 := M - M_3$ is a compact set which just has β-blockable line segments. For that set, Theorem 7.5 can be applied which yields the existence of a suitable blocking set. ■

Corollary 7.9 (Existence of β-EGs with length-bounded edges) *Let S be a compact surface without boundary and with continuous normals, which also may have non-blockable line segments, and P be a finite set of sample points taken from S. For every $l_0 > 0$ and $\varepsilon > 0$, P can be extended to a finite set P' so that the edges $e = \overline{\mathbf{pq}}$ of the corresponding β-environment graph have at least one of the following properties:*

(1) $l(e) \leq l_0$

(2) $\mathbf{n}(\mathbf{p})^T(\mathbf{p}-\mathbf{q}) > ||\mathbf{n}(\mathbf{p})|| \cdot ||\mathbf{p}-\mathbf{q}|| \cdot (1-\varepsilon)$,
$\mathbf{n}(\mathbf{q})^T(\mathbf{p}-\mathbf{q}) > ||\mathbf{n}(\mathbf{q})|| \cdot ||\mathbf{p}-\mathbf{q}|| \cdot (1-\varepsilon)$.

Proof: The corollary is an immediate implication of Corollary 7.8. ∎

An implication of these investigations is that non-β-blockable line segments cannot be avoided as bridge edges of the β-EG, cf. Section 5.2. However, for compact surfaces a positive infimum $l_{\min}$ of the length of its non-β-blockable line segments exists. The reason is that the set of β-blockable line-segments is open. We choose l_0 of Corollary 7.8 significantly smaller than $l_{\min}$. For the sampling sets P' belonging to l_0 according to the Corollary, the edges of the β-EG of P' not close to a non-β-blockable line segment of the surface are significantly shorter than those which are close and thus occur as bridge-edges in the graph. If we assume that there are no vertices with just bridge edges as incident edges, the clustered environment graph of P' should not contain any bridge edges. We summarize this discussion as an observation.

Observation 7.10 (Short-edge property of the clustered β-EG) *Let S be a compact SF-surface without boundary, P a finite set of points on S. An $l_{\min} > 0$ exists so that, for all $l_0 < l_{\min}$, P can be extended to a finite point set P' for which the clustered β-environment graph, $0 \leq \beta \leq 1$, of P' should only have edges of length less than l_0. The same holds trivially for every finite extension of P' by points on S.*

7.2 Intersection-free Embeddings

As we have noticed in the last chapter, the β-environment graph of a set of points in the plane is a planar graph for $\beta \geq 0$. This is a consequence of the property of being a subgraph of the Delaunay triangulation, but it can also be derived from a condition of intersection freeness which we give in the following. We formulate the condition for curved surfaces from which the planar case is obtained by specialization.

Definition 7.11 (Surface diameter disc) *Let S be a surface, and k be a shortest curve on S between two points $\mathbf{p}, \mathbf{q} \in S$. Let $\mathbf{r}$ be the point on the curve k with equal surface distance $l/2$ to $\mathbf{p}$ and $\mathbf{q}$, l the length of k. Then the closed **surface diameter disc** of k is the set of all points with surface distance from $\mathbf{r}$ less than or equal to $l/2$.*

Theorem 7.12 (Empty disc intersection criterion) *Let k, k' be two shortest curves with end points $\mathbf{p}, \mathbf{q}$ and $\mathbf{p}', \mathbf{q}'$, respectively, on a surface so that each closed surface diameter disc does not contain the end points of the other curve. Then the curves do not intersect.*

Proof: The proof is by contradiction, that is, we assume that k and k' intersect each other. Let $\mathbf{r}'$ be one of the intersection points, cf. Figure 7.2. Let $\mathbf{p}'$ be w.l.o.g. the end point of shortest distance to $\mathbf{r}'$. Let

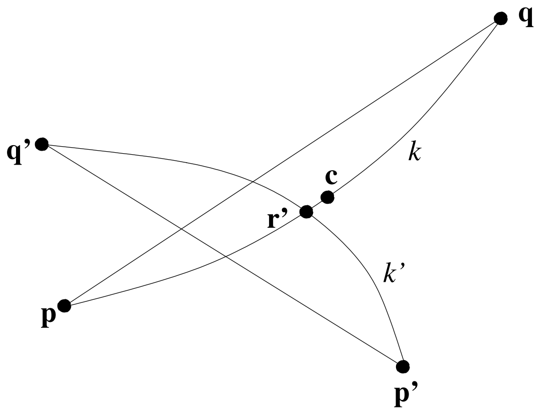


Figure 7.2: Configuration of intersection of two surface curves.

p w.l.o.g. be the end point of shortest distance to $\mathbf{r}'$ on the other curve k, that is, $d_S(\mathbf{p}', \mathbf{r}') \leq d_S(\mathbf{p}, \mathbf{r}')$. Let **c** be the point on k with equal surface distance to **p** and **q**. Then

$$d_S(\mathbf{p}', \mathbf{c}) \leq d_S(\mathbf{p}', \mathbf{r}') + d_S(\mathbf{r}', \mathbf{c}) \leq d_S(\mathbf{p}, \mathbf{r}') + d_S(\mathbf{r}', \mathbf{c}) = d_S(\mathbf{p}, \mathbf{c}),$$

where $d_S(.,.)$ denotes the distance on the surface.

That means that $\mathbf{p}'$ lies in the surface diameter disc of k which contradicts to the assumption of the theorem. ■

In the case of reconstruction, we do not know the surface. We just see the points which can be interconnected by line segments. In that setting, the following definition is useful.

Definition 7.13 (Conflict-free line segments) *Two line segments $s = \overline{\mathbf{pq}}$, $s' = \overline{\mathbf{p}'\mathbf{q}'}$ are called **conflict-free** if each (closed) diameter ball does not contain the end points of the other line segment.*

For the β-environment graphs with $\beta \geq 0$ all edges are conflict-free in that sense. The key question is whether it is possible to conclude intersection-freeness of an embedding, in the sense of the following definition, from conflict-freeness of a pair of line-segments.

Definition 7.14 (NN-Intersection) *Let S be a surface and $\mathbf{f}$ be a mapping which defines an embedding $\mathbf{f}(s)$ of a line segment s into S. Two line segments s, s' with vertices on S are called **intersection-free under** $\mathbf{f}$ if the curves $\mathbf{f}(s)$ and $\mathbf{f}(s')$ are intersection-free. If $\mathbf{f}$ is the NN-embedding, the line segments are called **NN-intersection-free**.*

Unfortunately, it seems hard or even impossible to find a sampling set for a given embedding, like e.g. the NN-embedding for which the edges of an environment graph all are intersection-free in that sense. A hint is that e.g. two edges of equal length in a common plane which intersect at their centers become conflict-free even if one of them is just slightly moved in space so that the intersection point is dissolved.

A way out consists in identification and elimination of critical edges. We suggest two approaches. The first one is based on a stronger definition of conflict-freeness, the so-called *a-conflict-freeness*, from which NN-intersection-freeness can be concluded for sufficiently short edges. The second one is a heuristic characterization of intersection, called χ*-intersection* which is also useful in the second part of the algorithm.

7.2.1 The Concept of a-Conflict-Freeness

The idea of a-conflict-freeness is to slightly enlarge the environment which has to be point-free.

Definition 7.15 (a-conflict-freeness) *Let $s = \overline{\mathbf{pq}}$, $s' = \overline{\mathbf{p'q'}}$ be two line segments between points $\mathbf{p}, \mathbf{q}, \mathbf{p'}, \mathbf{q'}$ on an SF-surface S within a save r-fringe of S, and $a : S \times S \to \mathbb{R}_+$ be a non-negative function. s and s' are called* ***a-conflict-free*** *if the enlarged closed diameter balls with radius $\frac{1}{2}l(s) + a(s) + a(s')$ and center $\frac{1}{2}(\mathbf{p}+\mathbf{q})$, and radius $\frac{1}{2}l(s') + a(s') + a(s)$ and center $\frac{1}{2}(\mathbf{p'}+\mathbf{q'})$, respectively, do not contain the end points of the other line segment.*

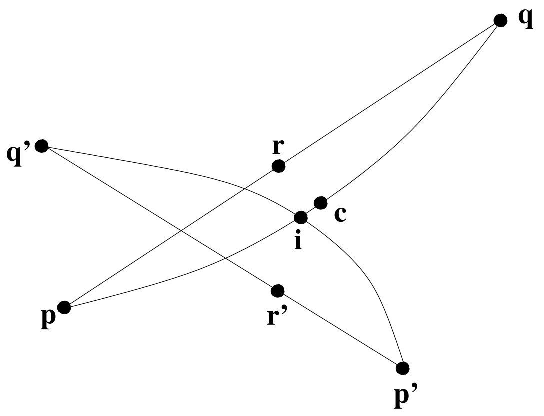


Figure 7.3: The case for the intersection freeness of Theorem 7.16.

The usefulness of the definition is demonstrated by the following theorem.

Theorem 7.16 (Intersection freeness) *Let S be a compact SF-surface without boundary and* **f** *be the NN-mapping. Let $s = \overline{\mathbf{pq}}$, $s' = \overline{\mathbf{p'q'}}$ be two line segments between points $\mathbf{p}, \mathbf{q}, \mathbf{p'}, \mathbf{q'}$ on S completely contained in a save fringe of S, $h(s)$ and $h(s')$ the largest distances of s and s' from S, respectively, and $a(s) \geq h(s)$, $a(s') \geq h(s')$. If s and s' are a-conflict-free then the embeddings $\mathbf{f}(s)$ and $\mathbf{f}(s')$ do not intersect.*

Proof: By Theorem 6.5, $\mathbf{f}(s)$ and $\mathbf{f}(s')$ are curves. We assume that $\mathbf{f}(s)$ and $\mathbf{f}(s')$ intersect each other. Let $\mathbf{r}$ and $\mathbf{r'}$ be points on s and s' which map under $\mathbf{f}$ to an intersection point $\mathbf{i}$. Let w.l.o.g. $\mathbf{p}$ be the end point of shortest distance on s to $\mathbf{r}$, and $\mathbf{p'}$ the end point of shortest distance on s' to $\mathbf{r'}$. Let w.l.o.g. $\mathbf{p'}$ be the shorter one of these two distances, that is, $||\mathbf{p'} - \mathbf{r'}|| \leq ||\mathbf{p} - \mathbf{r}||$. Let $\mathbf{c}$ be the point on s with equal distance to $\mathbf{p}$ and $\mathbf{q}$, cf. Figure 7.3. Then

$$\begin{aligned}
||\mathbf{p'} - \mathbf{c}|| &\leq ||\mathbf{p'} - \mathbf{r'}|| + ||\mathbf{r'} - \mathbf{c}|| \\
&\leq ||\mathbf{p} - \mathbf{r}|| + ||\mathbf{r'} - \mathbf{r}|| + ||\mathbf{r} - \mathbf{c}|| \\
&\leq ||\mathbf{p} - \mathbf{r}|| + ||\mathbf{r'} - \mathbf{i}|| + ||\mathbf{i} - \mathbf{r}|| + ||\mathbf{r} - \mathbf{c}|| \\
&\leq ||\mathbf{p} - \mathbf{c}|| + h(s) + h(s') \\
&= \frac{l(s)}{2} + h(s) + h(s') \\
&\leq \frac{l(s)}{2} + a(s) + a(s').
\end{aligned}$$

That means that $\mathbf{p'}$ lies in the enlarged diameter sphere of s, and thus s and s' are not a-conflict-free, in contradiction to the condition of the theorem. ∎

The following definition combines the concept of a-conflictness with β-environments.

Definition 7.17 (**a-β-environment graph**) *Let S be a surface and $a : S \times S \to \mathbb{R}_+$ be a non-negative function, P a finite set of points on S. The graph with vertex set P and line segments $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in P$, as edges, for which the environments $E'_{\beta,a}(a) := E_\beta(s) \cup E(s,a)$ with $E(s,a)$ the ball centered at the midpoint $\frac{1}{2}(\mathbf{p}+\mathbf{q})$ of s and radius $r'(s) = \frac{1}{2}l(s) + 2a(s)$, do not contain any point of P, is called a-β-**environment graph**. $E'_{\beta,a}(a)$ is denoted as a-β-**environment**.*

We now derive arguments which show that for a-β-environment graphs, or subgraphs of them like clustered a-β-environment graphs which can be defined immediately, with sufficiently short edges, the ratio $a(s)/l(s)$ for $a(s)$ guaranteeing intersection-freeness can be made small.

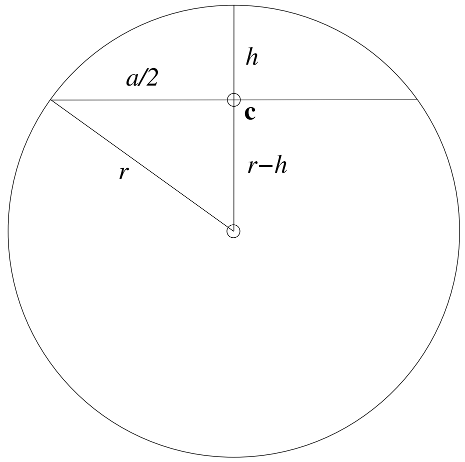


Figure 7.4: Calculation of the distance h of a chord of length a to the circle.

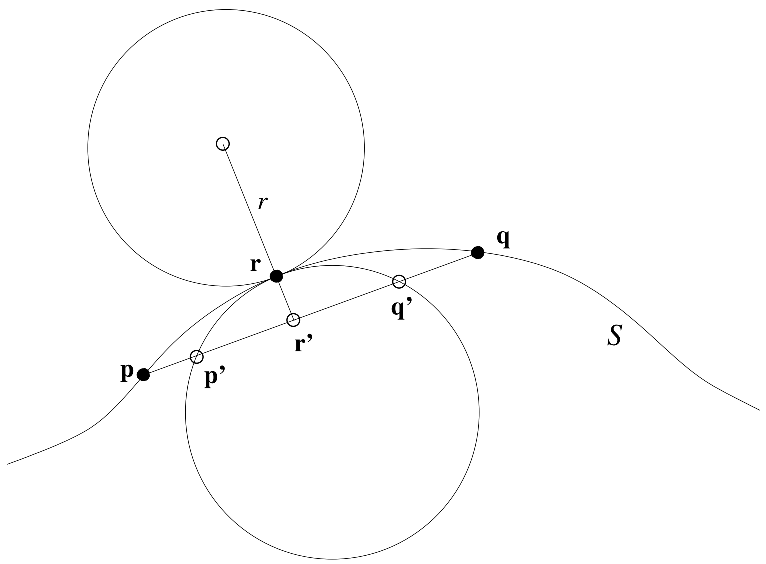


Figure 7.5: Estimation of the distance of a line segment $\overline{\mathbf{pq}}$ from a surface S. The figure shows the configuration in the slicing plane spanned by $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$.

Lemma 7.18 *We consider a circle of radius r, a chord of length $a = 2\alpha \cdot r$, $0 < \alpha < 1$, and the maximum distance h of a point on the chord from the circle. Then $\lim_{a \to 0} \frac{h}{a^2} = \frac{1}{8r}$.*

Proof: The maximum distance is reached at the center $\mathbf{c}$ of the chord, cf. Figure 7.4. The point $\mathbf{c}$, an endpoint of the chord, and the center of the circle define a rectangular triangle whose edge lengths satisfy

$$(r-h)^2 - (\alpha r)^2 = r^2,$$

or, resolved for h,

$$h = r(1 - \sqrt{1-\alpha^2}).$$

Then

$$\begin{aligned}
\lim_{a \to 0} \frac{h}{a^2} &= \lim_{\alpha \to 0} \frac{r(1-\sqrt{1-\alpha^2})}{4(\alpha r)^2} \\
&= \lim_{\alpha \to 0} \frac{r(1-\sqrt{1-\alpha^2})(1+\sqrt{1-\alpha^2})}{4(\alpha r)^2(1+\sqrt{1-\alpha^2})} \\
&= \lim_{\alpha \to 0} \frac{r\alpha^2}{4(\alpha r)^2(1+\sqrt{1-\alpha^2})} \\
&= \frac{1}{8r}.
\end{aligned}$$

■

Theorem 7.19 (Estimation of maximum surface distance) *Let S be a compact SF-surface without boundary with an SF-radius r, and $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in S$, a line segment within the r-fringe, $h(s)$ the maximum distance of a point of s from S. Then*

$$\overline{\lim}_{l(s) \to 0} \ \frac{h(s)}{l(s)^2} < \frac{1}{8r}.$$

Proof: Let $\mathbf{r}'$ be a point on s with maximum distance to S, and $\mathbf{r}$ a surface point having that distance. We consider the triangle $\triangle(\mathbf{p}, \mathbf{r}, \mathbf{q})$, cf. Figure 7.5. Because the line through $\mathbf{r}'$ and $\mathbf{r}$ is perpendicular to S, the plane E spanned by $\triangle(\mathbf{p}, \mathbf{r}, \mathbf{q})$ is a normal section plane, that is, it contains the normal vector of S at $\mathbf{r}$. The consequence is that the intersection of E with the two tangent balls at $\mathbf{r}$ with radius r are two discs with radius r. The endpoints of s are outside of the discs. Since we are interested in the limit, we can assume that the length $l(s)$ of s is less than $2\alpha r$ for some α between 0 and 1. By construction, $\mathbf{r}$ is between $\mathbf{p}$ and $\mathbf{q}$ on the intersection curve $S \cap E$. s intersects at least one of the two discs. We take that chord s' whose intersection points $\mathbf{p}', \mathbf{q}'$ with the boundary of the disc are on different sides of $\mathbf{r}$ on the boundary, or equivalently contains $\mathbf{r}'$. As a subsegment of s, the length of s' is less than $2\alpha r$, too.

If the line segment $\overline{\mathbf{rr}'}$ is perpendicular to s', we can immediately apply the previous lemma. Otherwise, $l(\overline{\mathbf{rr}'}) \leq h$, h the distance of $\mathbf{r}'$ from the surrounding circle. Hence by the preceding Lemma 7.18, the estimation of the limit holds for that case, too. ■

Now we can prove the main theorem of this section.

Theorem 7.20 (Intersection-freeness of α-β-environment graphs) *Let S be a compact SF-surface without boundary, $r > 0$ an SF-radius of S. For every $\varepsilon > 0$, an $l_\varepsilon > 0$ exists so that all graphs G which*

(1) are subgraphs of an a-β-environment graph, $\beta \geq 0$, $a(s) = (\frac{1}{8r} + \varepsilon) \cdot l(s)^2$, with vertices on S,

(2) are completely inside the r-fringe of S,

(3) have a maximum edge length less than l_ε,

are intersection-free, that is, the NN-embeddings of the edges of G do not intersect.

Proof: By Theorem 7.16, intersection-freeness can be concluded from a-conflict-freeness for the edges of a graph G if (2) is satisfied for G, and $a(s) \geq h(s)$. From Theorem 7.19 we know that for each $\varepsilon > 0$ an $l_\varepsilon > 0$ exists so that $h(s) < (1/(8r) + \varepsilon) \cdot l(s)^2 = a(s)$ for $l(s) < l_\varepsilon$. Thus, intersection freeness of the edges of G can be concluded from a-conflict-freeness if (1), (2), and (3) hold for the edges of G. ■

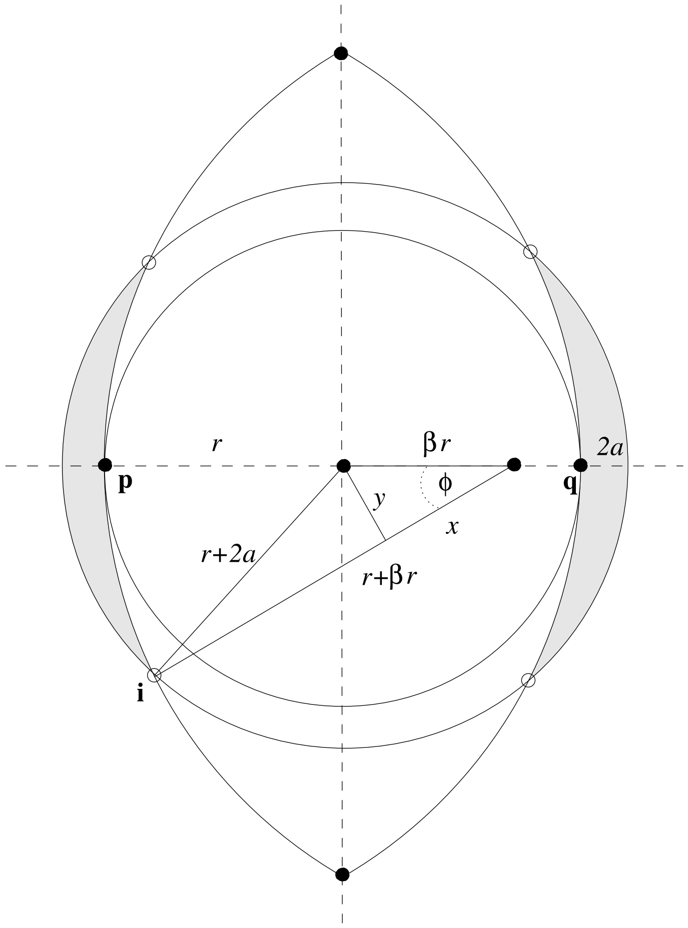


Figure 7.6: Illustration of the difference between $E(s, a)$ and $E_\beta(s)$. The difference set $D_{\mathbf{p}}(s)$ is drawn shaded.

The theorem tells that for β-environment graphs with sufficiently short edges, a tolerance of second order in the edge length for the emptiness of the β-environments is sufficient in order to have intersection-freeness of the nearest-neighbor embedding. This implies that intersection-freeness of edges can be expected as the usual case for β-environment graphs with short edges.

The a-β-environment of a line segment s is a superset of the β-environment of s. The difference is $D_\beta(s, a) := E(s, a) - E_\beta(s)$. With increasing β, $D_\beta(s, a)$ decreases. The following theorem analyzes this behavior quantitatively.

Theorem 7.21 (Difference between a-β- and β-environments) *Let $s = \overline{\mathbf{pq}}$ be a line segment, $l(s)$ its length, $a(s) < \frac{1}{4}(\sqrt{1+2\beta}-1)l(s)$, $D_\beta(s,a) := E(s,a) - E_\beta(s)$, $E(s,a)$ the ball centered at the midpoint $\frac{1}{2}(\mathbf{p}+\mathbf{q})$ of s and radius $r'(s) = \frac{1}{2}l(s) + 2a(s)$. Then the following holds:*

(1) $D_\beta(s,a)$ consists of two connected components $D_\mathbf{p}(s)$ and $D_\mathbf{q}(s)$ which are incident to $\mathbf{p}$ and $\mathbf{q}$, respectively.

(2) The maximum distance of a point in $D_\mathbf{p}(s)$ from $\mathbf{p}$ is smaller than the upper bound $d_{\max}(s)$ which satisfies

$$d_{\max}(s) = (1+\beta)\ \frac{1}{2}l(s)\ \phi(s),$$

where

$$\phi(s) := \arcsin\left(2\sqrt{\frac{1}{(1+\beta)\beta}}\sqrt{\frac{a}{r}}\left(1+\mathcal{O}\left(\frac{a}{r}\right)\right)\right) \quad \text{for} \quad 0 < \beta \leq 1.$$

The same holds for $D_\mathbf{q}(s)$ and $\mathbf{q}$.

(3) For $a(s) := (\frac{1}{8r}+\varepsilon)\cdot l(s)^2$ for some constants $r > 0$, $\varepsilon > 0$,

$$\lim_{l(s)\to 0} \frac{d_{\max}(s)}{l(s)} = 0 \ \text{ for } \ \text{for } 0 < \beta \leq 1.$$

That means that the difference between the $a(s)$-β-environment and the β-environment can be made arbitrarily small.

Proof: Let $r := \frac{1}{2}l(s)$. We consider a plane through the center point of s. Figure 7.6 depicts the intersection of the spatial configuration with that plane. The boundaries of the environments are arcs. For $E(s,a)$ the boundary is in fact a circle of radius $r+2a$, while the boundary of $E_\beta(s)$ consists of two symmetric arcs of radius $(1+\beta)r$. These arcs intersect the circle if their common points are outside the circle if

$$\sqrt{(1+\beta)^2r^2 - \beta^2r^2} > r + 2a.$$

Because this holds by the condition $a(s) < \frac{(\sqrt{1+2\beta}-1)l(s)}{4}$ on $a(s)$ of the theorem, (1) is satisfied.

In order to prove (2), let us consider Figure 7.6. The circle of $E(s,a)$ and the two arcs of $E_\beta(s)$ intersect in four points. The difference $D_\mathbf{p}$ (and analogously the difference $D_\mathbf{q}$) is located between two arcs induced by the appropriate pair of intersection points. The maximum distance $d_{\max}(s)$ is bounded by the length of the arc c of $E_\beta(s)$ between $\mathbf{p}$ and one of these intersection points, denoted by $\mathbf{i}$. We calculate the length of arc c.

For that purpose we determine the angle ϕ defining the arc as part of its circle. The cosine of ϕ is given by

$$\cos(\phi) = \frac{x}{\beta r},$$

where $r := l(s)/2$ and x satisfies the relations

$$x^2 + y^2 = (\beta r)^2,$$

$$(r + \beta r - x)^2 + y^2 = (r+2a)^2,$$

and y is defined as shown in Figure 7.6. From those two equations we get

$$x = \beta r - \frac{2a}{1+\beta} - \frac{2a^2}{(1+\beta)r},$$

and thus

$$\cos(\phi) = 1 - \frac{2a}{(1+\beta)\beta r} - \frac{2a^2}{(1+\beta)\beta r^2} = 1 - k(\beta)\frac{2a}{r} - k(\beta)\frac{2a^2}{r^2},$$

with

$$k(\beta) := \frac{1}{(1+\beta)\beta}.$$

For the sine we get

$$\sin(\phi) = \sqrt{1-\cos^2(\phi)} =$$

$$\sqrt{k(\beta)\,\frac{4a}{r} + (k(\beta)-k(\beta)^2)\,\frac{4a^2}{r^2} - k(\beta)^2\,\frac{8a^3}{r^3} - k(\beta)^2\,\frac{4a^4}{r^4}},$$

and thus

$$\sin(\phi) < \sqrt{k(\beta)\,\frac{4a}{r} + k(\beta)\,\frac{4a^2}{r^2}} = 2\sqrt{k(\beta)}\sqrt{\frac{a}{r}}\left(1 + \mathcal{O}\left(\frac{a}{r}\right)\right).$$

Because the length of the arc c is $l(c) = 2r\phi$, and because $r = \frac{l(s)}{2}$, we get assertion (2) of the theorem.

(3) can be immediately proved by putting $a(s)$ into the formulas of (2). ∎

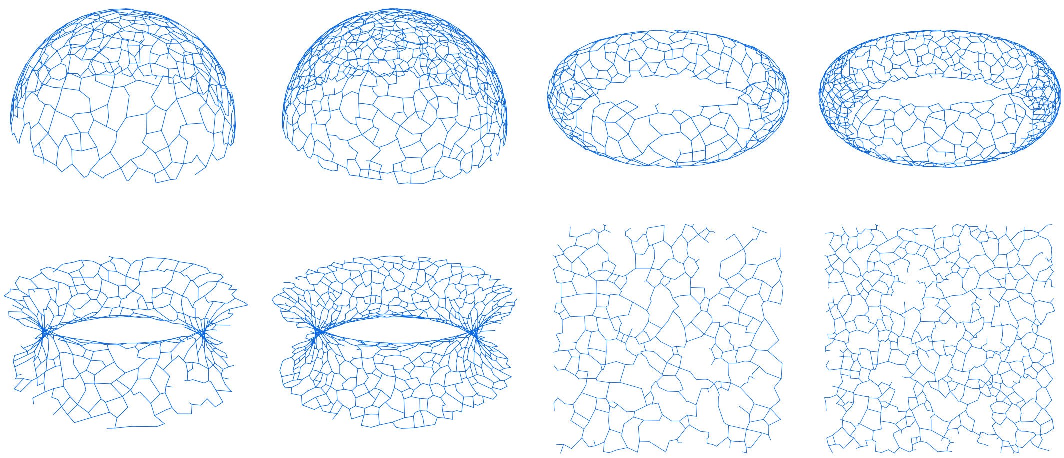


Figure 7.7: The test examples with random sampling.

The investigations can be summarized in the following observation.

Observation 7.22 (Intersection-freeness of clustered β-EGs) *A consequence of statement (3) of Theorem 7.21 and Theorem 7.20 is that clustered β-environment graphs, $0 < \beta \leq 1$, can be made arbitrarily close to intersection-freeness. Just pairs of edges for which a vertex of one of them is very close to a vertex of the other one might intersect in the NN-image. But the chance that this happens can be made arbitrarily small if the maximum edge length decreases to 0, and it decreases if β increases.*

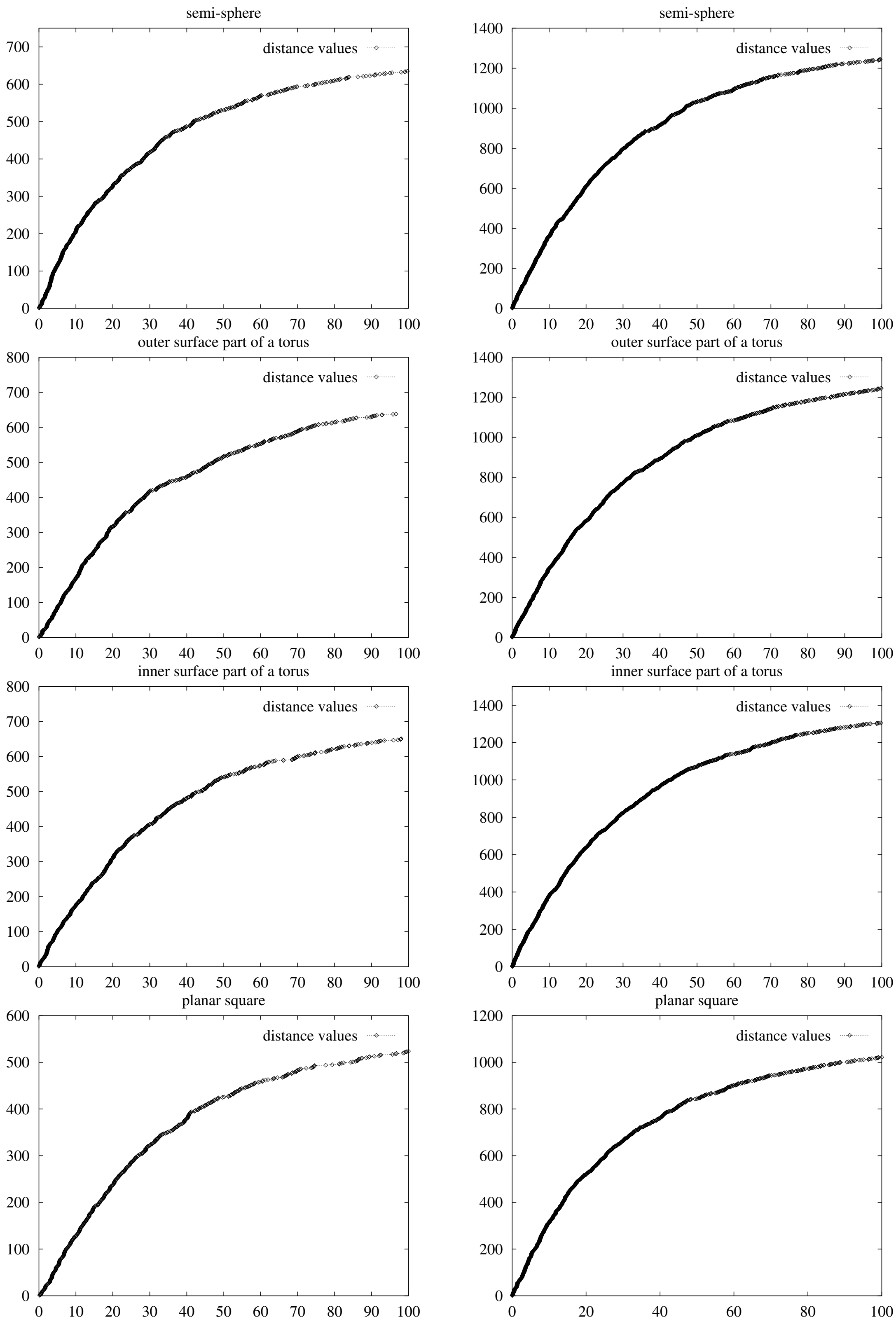


Figure 7.8: Empirical analysis for the randomly sampled objects (lower point density on the left and higher density on the right) of the shortest distance of a point to the 1-environment of any edge of the clustered 1-EG. Only distances up to 100 percent of the length of the considered edge are displayed.

In addition to the theoretic investigations an empirical analysis of the shortest distance of a point to the β-environment of an edge of the clustered β-EG for $\beta = 1$ has been performed on representative smooth surfaces of different types of curvature: a semi-sphere, the inner and outer part of a torus, and a planar square. The sample points are chosen randomly at two different densities. The number of sample points of the case of higher density is about twice of the number of the case of lower density. Figure 7.7 shows the clustered β-environment graphs for those sample sets.
The samples on the semi-spheres are obtained by determining approximately equal-spaced points located on circles in parallel to the equator. This implies that the number of points on a circle decreases with the perimeter. Then the points are jittered by modifying their positions by random offsets along their circles and perpendicular to it. The sample sets of the outer (and analogously of the inner) region of the tori are analogously generated by arranging approximately equal-spaced points on circles in parallel to the "equator" of the torus and covering the outer region. Then the points are jittered by modifying their positions by random offsets along their circle and perpendicular to it. The square is randomly sampled with the required number of points by a random function delivering x- and y-coordinates scaled to the side length of the square.
The curves shown in Figure 7.8 represent the number of points for which the ratio between the shortest distance of any other point and the edge length is at most equal to the percentage given on the horizontal axis. The plot is restricted to distances up to 100% of the edge length. We can note that the behavior of the curves is independent from the density of the sampling set if the number of points is taken relatively to the total number of sample points.

7.2.2 Intersecting Edges

For the concept of χ-intersecting line segments of Definition 5.14 in Chapter 5 we have demonstrated in Theorem 5.15 that this concept is equivalent to the notion of intersection if the line segments are located in the plane. In space, the definition is reasonable, too.

Theorem 7.23 *Let s_1 and s_2 be not χ-intersecting for $\chi > 90°$. Furthermore, let $0° < \gamma_{\min} < \gamma_{\max} - \gamma_{\min} < 180°$, and the four vertices of s_1 and s_2 so that their distances are shorter than r_0 of Theorem 6.16 of Chapter 6. Let t_1 and t_2 be two triangles of (1) of Definition 5.14 with a dihedral angle of at least χ.*
If t_1 and t_2 each have a vertex with angle γ between $\gamma_{\min}$ and $\gamma_{\max}$, then the NN-images of s_1 and s_2 do not intersect.

Proof: Theorem 6.16 implies that then the NN-images of t_1 and t_2 do not intersect, and thus s_1 and s_2 which are a non-common subsets of them do not, too. ■

Otherwise three of the vertices of s_1 and s_2 define a slim triangle with angles outside the interval $(\gamma_{\min}, \gamma_{\max})$. For a large $\gamma_{\max}$ this means that three vertices are almost co-linear what should not happen too often.
If condition (2) of Definition 5.14 does not hold, then the NN-images of the two adjacent triangles should intersect, and thus the vertices of the second line segment should "lie on the same side" of the first line segment. Thus the two line segments should not have an intersection point.
A further investigation of the possible mutual locations of s_1 and s_2, omitted here, might yield further constraints which reduce the probability of configurations of NN-intersection further which cannot be decided by the criterion of χ-intersection.
Table 7.1 shows the result of an empirical investigation of the clustered β-environment graphs of our examples for $\beta = 0, \frac{1}{2}$ and 1, and $\chi = 90°$. We can notice that even for this generous bound, the number of χ-intersecting edges is usually neglectably small, in particular for higher values of β. In fact, for the values $\beta \in \{\frac{1}{2}, 1\}$ even no intersection did occur.

	# of eliminated intersecting edges in the clustered β-EG							
	torus	cup	head	skull	puppet	cap	pharaoh	tori
# points	310	2650	1487	698	695	371	2286	620
$\beta = 1$	0	0	0	0	0	0	0	0
$\beta = \frac{1}{2}$	0	0	0	0	0	0	0	0
$\beta = 0$	0	275	0	8	5	3	16	1

Table 7.1: The number of χ-intersecting edges of the clustered β-environment graphs, $\beta \in \{0, \frac{1}{2}, 1\}$, for the example data sets.

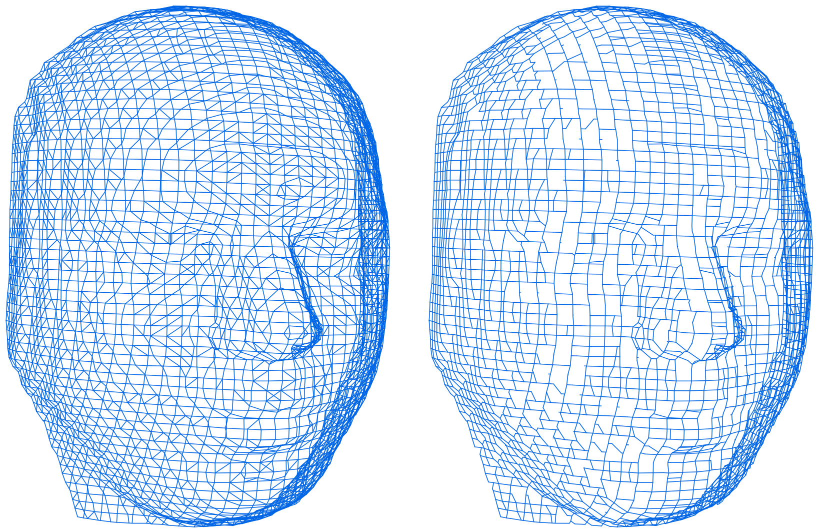


Figure 7.9: The 0-environment graph and the 1-environment graph of a surface scanned by the MC-algorithm.

7.3 Discussion

The investigations of this chapter can be summarized in the following observation.

Observation 7.24 (Usefulness of clustered β-EGs) *Let S be a compact SF-surface without boundary. By Observations 7.10 and 7.22, applied with an edge length bound fulfilling the requirements of Theorem 6.5, a given sample set P on S can be extended to a finite point set P' so that, with high probability, the clustered β-EG, $0 \leq \beta \leq 1$, of P' should be NN-embeddable into S. Any finite extension of P' by points on S does have this property, too.*

This observation makes just an assertion on the existence of favorable sample sets, but how should sample sets be chosen in practice? Our experiments with uniform-random sample sets presented in this section show that this type of sampling is acceptable for the reconstruction algorithm. A reason is that with increasing sample density, the length of edges of the β-EG reduces because the chance that a sample point falls into the β-environment of a long line segment increases.

The sample points need not to be distributed randomly. A grid-like surface sampling with uniformly sampled points is at least as well suited. For surfaces not containing flat or umbilical points, lines of minimum and maximum curvature may be used to define a mesh of curves which meet orthogonally in their intersection points. If the mesh is chosen dense, the approximation of the curve segments

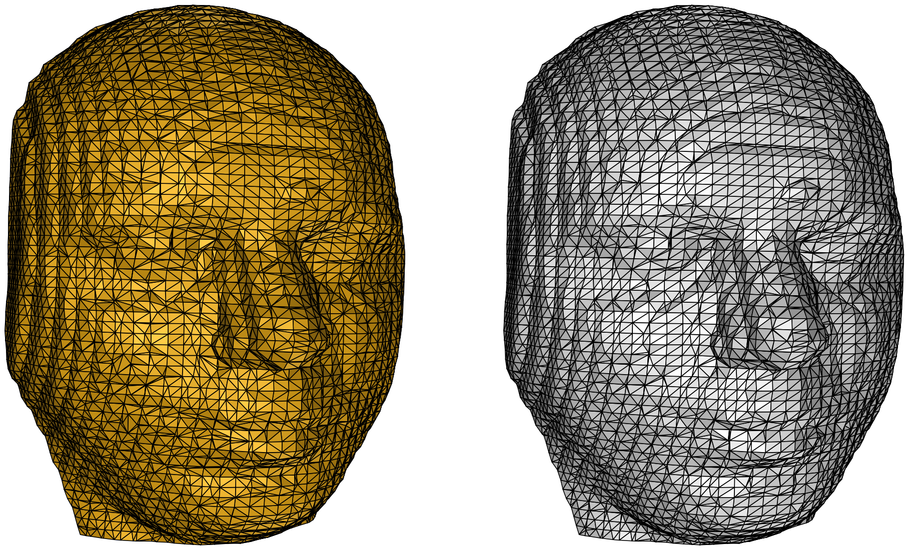


Figure 7.10: Left: the reconstruction out of the 1-environment graph. Right: the reconstruction with the marching cubes algorithm.

between two consecutive intersection points by line segments may induce a mesh of loops with angles of about 90° at two adjacent edges. The loops of such a mesh are usually quadrilaterals, and hence have a constant number of edges. Furthermore, the diameter discs of the edges should in many cases be empty of vertices, so that the 0-environment graph should contain many of them. Although the restrictions for the 1-environment graph are higher than for the 0-environment graph, this can also be achieved for this type of graph, as it can be seen in the example of the cup in Figure 5.9 of Chapter 5.

Another observation is that the edges delivered by the marching cubes algorithm from a spatial grid sampling of a surface, are a subset of edges reconstructed by the 0-environment graph. The marching-cubes (MC) algorithm [LC87] samples closed surfaces by a rectangular spatial grid. The vertices of the grid are classified as interior or exterior with respect to the surface. From the vertex classification of a cube of the mesh, a configuration of intersection of the surface with the cube is derived which consists of one or more surface loops. These loops, or a triangulation of them are reported as a surface approximation.

The MC-algorithm can also be seen as an edge-scan algorithm. The edges of the grid intersecting the surface are determined. A sampling grid is proper if every edge contains at most one intersection point with the surface. These intersection points define the loop configuration of a cube. The advantage of that view is that the surfaces need neither be closed nor orientable. If the sampling is restricted to cubes not traversed by the boundary of the surface, the MC algorithm yields an approximation of the surface.

For SF-surfaces, due to their bounded curvature, the sampling grid of the marching cubes algorithm can be chosen so dense that configurations with more than two sample points on an edge of a square do not occur. Furthermore, the sampling grid can be chosen so dense that MC-configurations with disconnected traversals or tubes do not occur, that is, every cube defines a single loop.

The diameter disc of an edge in the scan square in which the edge is located does not contain any sampling point. The reason is that it only may intersect edges of the square, but those edges do not contain any other sampling points than those inducing the edge, due to the chosen sampling grid with just one surface traversal.

Further, the diameter disc of an edge in the scan square of the edge does not contain any of the vertices of the square in its interior. Thus the diameter ball does not intersect any edge of the scan grid perpendicular to the square. Hence the diameter ball does not contain any sampling point in its interior. Thus the edge is in the 0-environment graph.

Hence if the intersection curves with the plane are scanned by the grid on that plane so that all edges have an empty diameter circle, all edges of the marching cubes algorithm belong to the 0-environment graph.

The number of edges on a surface loop of an MC-mesh is at most seven. That means that we have the desired property of surface loops with a bounded number of edges.

By increasing the resolution of the sampling grid, the sampling points can be made arbitrarily dense on the surface. That means that the clustered environment graph has a good chance not containing non-blockable bridge edges if the density is chosen sufficiently high.

Figure 7.9 shows the 0-environment graph and the 1-environment graph of a surface scanned by the MC-algorithm. The reconstruction result out of the 1-environment graph is depicted in Figure 7.10.

For β-environment graphs with $\beta > 0$ the argument concerning the emptiness of the diameter ball of an edge does not hold. This means that not all MC-edges belong to the graph. In the case of $0 < \beta \leq 1$ a scan grid with regular triangles as faces might help. A difficulty in finding a suitable scan grid is that the angle between a face and an incident non-face edge should be at least $90°$ in order that the β-environment does not intersect edges incident to the face of the considered edge.

If we summarize our investigations, we can conclude that a choice of $0 \leq \beta \leq 1$ is reasonable for an appropriately sampled surface. However, for $\beta < 1$ slightly stronger restrictions on the quality of the surface sampling have to be made than for $\beta = 1$. The depicted examples in Figure 5.9 of Chapter 5 show that a value of $\beta = 1$ adapts very well to high point density changes as well as to regions with strong curvature. A reason is that for $\beta = 1$ the clustered β-environment graph has the same desired reconstruction precision as the EMST, cf. Chapter 4. As consequence, if nothing is known on the surface sampling a value of $\beta = 1$ is a good choice for the clustered β-environment graph.

Chapter 8

Triangulation

This chapter describes the second phase of the reconstruction algorithm, the triangulation, together with new definitions required for its formulation.

8.1 The Algorithm

Algorithm 8.1 summarizes the approach of triangulation. The algorithm consists of two main steps, the generation of a partial embedding, indicated by (1), and a phase of incremental triangulation, indicated by (2). The goal of step 1 is the generation of very local embeddings of the surface description graph G obtained from phase 1 of the reconstruction algorithm. Each local embedding concerns a vertex and its incident edges in G. A local embedding of this type is achieved by defining a sorted arrangement of the incident edges of the vertex. The local surface into which the vertex and its edges are embedded is the "umbrella" of triangles obtained by closing every sector defined by two consecutive edges of the arrangement by a chord. Usually the union of those local embeddings will not yield a manifold surface, so that we do not necessarily use these triangles. A detailed investigation has to be performed in order to decide whether a triangle of this type is used as part of the reconstructed surface. This investigation is subject of step 2.

Step 2 processes the sectors obtained in step 1 according to a suitable priority. Depending on the opening angle and the location of points and edges in the environment of the sectors, the algorithm iteratively inserts new edges into graph G and adds triangles to the initially empty manifold M which finally defines the reconstructed surface, remember Figure 3.4 of Chapter 3.

The details of the two steps are described in the following sections.

8.2 Generation of a Partial Embedding

In the first step, the originally purely combinatorial graph is partially embedded. Embedding of a graph in principle means to find a surface in space of which the graph is a part. From combinatorial topology we know that an embedding is uniquely defined by the sorted arrangement of the incident edges at every vertex. Intuitively, if an embedding is given, two consecutive edges of the arrangement at a vertex lie on a common face of the surface into which the graph is embedded. If sorted edge arrangements at the vertices are given, the rule of face and thus of surface construction depends on whether the embedded manifold should be orientable or not [Whi73].

In combinatorial topology the goal usually is to find a topological embedding so that the genus of the embedding is minimized. In our setting, we can use the geometric information on the location of the vertices in space for the definition of the arrangements $\pi(\mathbf{v})$. The goal is to determine an optimally flat arrangement of the incident edges of a vertex. Flatness is in principle measured by considering

Algorithm 8.1 Triangulation of the surface description graph

Input: *The SDG G from phase 1 of the reconstruction algorithm.*
Parameters:

- *Line segment candidate region* $C_c(s)$.
- *Triangle candidate region* $C_c(t)$.
- *Boundary control angle* γ_c'.
- *Dihedral angle control* δ_c.
- *Line segment intersection control angle* χ_c.

Operation:

(1) For every vertex $\mathbf{p}$ *of G, determine an optimal dihedral arrangement of its incident edges.*
(2) Insert the sectors induced by the arrangement into a priority queue Q.

<u>**repeat**</u>
    Take the first sector $w = (\overline{\mathbf{pq}_1}, \overline{\mathbf{pq}_2})$ *from* Q *and remove it from* Q.
(A) <u>**if**</u> (*the angle of* w *at* $\mathbf{p}$ *is less than* γ_c') <u>**then**</u>
(B)   <u>**if**</u> (*a vertex* $\mathbf{q} \in C_c(t(w))$, $t(w)$ *the triangle induced by the sector* w, *exists*)
      <u>**then**</u>
        Insert $\overline{\mathbf{pq}}$ *into G where* $\mathbf{q}$ *is a vertex*
        for which $\overline{\mathbf{pq}}$ *does not intersect any edge of G.*
      <u>**else**</u>
(C)   <u>**if**</u> ($e_3 := \overline{\mathbf{q}_1\mathbf{q}_2}$ *is not an edge of G*)
        <u>**then**</u>
(D)   <u>**if**</u> (*a suitable point* $\mathbf{q} \in C_c(e_3)$ *different from* $\mathbf{p}$ *exists so that*
            the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ *does not fold-over and*
            $\overline{\mathbf{pq}}$ *does not intersect an edge of G*)
          <u>**then**</u>
            Determine the triangulation of the four points $(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q})$,
            for which the maximum angle of the triangles is minimized.
            <u>**if**</u> (*the newly introduced edge does not intersect an edge of G*)
            <u>**then**</u>
              Insert the new edge into G.
              <u>**if**</u> (e_3 *is the new edge of the triangulation*)
              <u>**then**</u> *insert triangle* $t(w)$ *into M.*
          <u>**else**</u>
            <u>**if**</u> (e_3 *does not intersect an edge of G*)
            <u>**then**</u> *insert* e_3 *into G; insert triangle* $t(w)$ *into* M.
        <u>**else**</u> *insert triangle* $t(w)$ *into* M*;*
<u>**until**</u> (*no more candidates can be found*).
Output: *A triangulation* M.

the dihedral angles between the induced triangles of two neighboring sectors of an arrangement. The following definition specifies the meaning of an optimal dihedral arrangement precisely.

Definition 8.1 (Optimal dihedral arrangement)
Let

- $G = (P, E)$ *be a geometric graph,*
- $\Pi(\mathbf{p})$ *be the set of all cyclic arrangements of the edges at vertex* $\mathbf{p}$ *where edges* $\overline{\mathbf{pq}}, \overline{\mathbf{pr}}$ *for which the edge* $\overline{\mathbf{qr}}$ *is in* E *are consecutive in the cycle of arrangement,*

- $180° > \delta_c > 0°$.

Then

- $Z(\pi(\mathbf{p}))$ *denotes the maximum number of consecutive edges so that the triangles induced by every subsequent pair of them have a dihedral angle in* $(\delta_c, 180°]$.
- $Z_{\max}(\mathbf{p})$ *denotes the maximum of* Z *over all cyclic arrangements in* $\Pi(\mathbf{p})$*:*
  $Z_{\max}(\mathbf{p}) := \max\{\ Z(\pi(\mathbf{p})) \mid \pi(\mathbf{p}) \in \Pi(\mathbf{p})\ \}$.
- $\mathcal{Z}_{\max}(\mathbf{p})$ *is the set of cyclic arrangements which achieve the maximum:*
  $\mathcal{Z}_{\max}(\mathbf{p}) := \{\ \pi(\mathbf{p}) \mid Z(\pi(\mathbf{p})) = Z_{\max}(\mathbf{p})$ *and* $\pi(\mathbf{p}) \in \Pi(\mathbf{p})\ \}$.
- $Q(\pi(\mathbf{p}))$ *denotes the sum of all dihedral angles between the triangles induced by every subsequent pair of edges.*
- $Q_{\max}(\mathbf{p})$ *is the maximum of* Q *over all arrangements maximizing* Z*:*
  $Q_{\max}(\mathbf{p}) := \max\{\ Q(\pi(\mathbf{p})) \mid \pi(\mathbf{p}) \in \mathcal{Z}_{\max}(\mathbf{p})\}$.
- $\mathcal{Q}_{\max}(\mathbf{p})$ *is the set of cyclic arrangements which achieve the maximum:*
  $\mathcal{Q}_{\max}(\mathbf{p}) := \{\ \pi(\mathbf{p}) \mid Q(\pi(\mathbf{p})) = Q_{\max}(\mathbf{p})$ *and* $\pi(\mathbf{p}) \in \mathcal{Z}_{max}(\mathbf{p})\ \}$.
- $V(\pi(\mathbf{p}))$ *denotes the variance of all dihedral angles between the triangles induced by every subsequent pair of edges.*
- $V_{\min}(\mathbf{p})$ *is the minimum of the variance* V *over all permutations that maximize* Q*:*
  $V_{\min}(\mathbf{p}) := \min\{\ V(\pi(\mathbf{p})) \mid \pi(\mathbf{p}) \in \mathcal{Q}_{max}(\mathbf{p})\ \}$.
- $\mathcal{V}_{\min}(\mathbf{p})$ *is the set of cyclic arrangements which achieve the minimum:*
  $\mathcal{V}_{\min}(\mathbf{p}) := \{\ \pi(\mathbf{p}) \mid V(\pi(\mathbf{p})) = V_{\min}(\mathbf{p})$ *and* $\pi(\mathbf{p}) \in \mathcal{Q}_{max}(\mathbf{p})\ \}$.

The arrangements in $\mathcal{V}_{\min}(\mathbf{p})$ *are called* ***optimal dihedral arrangements****.*

The reason for including the variance is to reduce the set of solutions if more than one optimal arrangement is found. It is not really necessary and can be omitted for the benefit of faster computation. Ties can be broken by arbitrary selection of one of the solutions.

Figures 8.1 and 8.2 show examples of optimal dihedral arrangements for a rather non-planar case of edges.

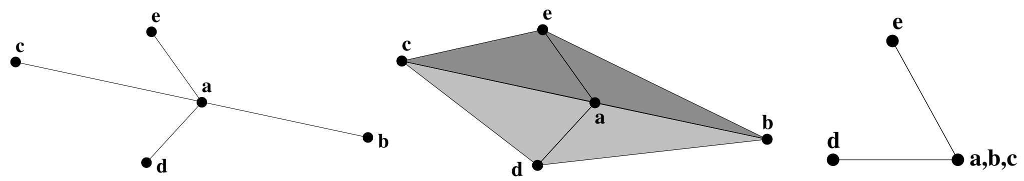


Figure 8.1: An example for an optimal dihedral arrangement (Definition 8.1) of a point at a sharp edge. Left: point **a** with its edges. Middle: the optimal dihedral arrangement. Right: view from side (**a**, **b**, **c** are collinear).

For graphs with vertex sets in the plane, this definition implies the canonical planar embedding.

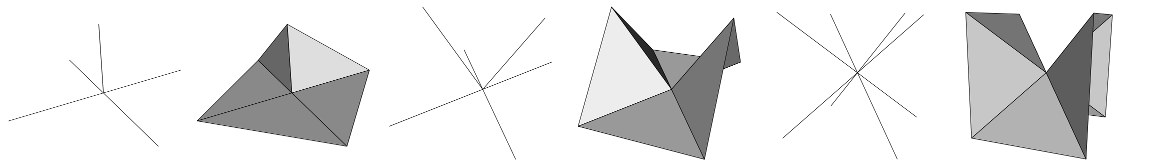


Figure 8.2: Examples of optimal dihedral arrangements for five, six, and eight incident edges to a point $\mathbf{p}_i$.

For vertices of degree higher than 3 there are two possibilities of orientation of the cyclic arrangement: clockwise and counter-clockwise. For a topological embedding, one of them has to be chosen. In our algorithm, that task is performed later-on as a side-effect of triangulation. For that reason we call the result of step 1 a *partial embedding*.

The search for an optimal dihedral arrangement can be performed by systematic enumeration. Because of the usually small vertex degrees of the used SDGs (Chapters 4 and 5) this approach is sufficiently efficient.

In the second phase of the algorithm, further edges may occur at a vertex, and it might happen that the resulting number of edges may increase computation time. Algorithm 8.2 achieves a speed-up by inserting so-called non-1-environment sectors around the considered point $\mathbf{p}$ before it computes the optimal dihedral arrangement according to Definition 8.1.

Definition 8.2 (Non-1-environment sector) *Let* $\mathbf{p}$ *be the center of a sector* w *with edges* $e_1 := \overline{\mathbf{pq}_1}$ *and* $e_2 := \overline{\mathbf{pq}_2}$. w *is a* ***non-1-environment sector*** *if* $\mathbf{q}_1$ *is in the 1-environment of* $\overline{\mathbf{pq}_2}$ *or* $\mathbf{q}_2$ *is in the 1-environment of* $\overline{\mathbf{pq}_1}$ *in the sense of Definition 5.8.*

This approach is reasonable, because the induced triangles of these non-1-environment sectors, or some kind of re-triangulation of them, can be already considered as a part of the correct surface mesh. If the edges of each non-1-environment sector would not belong to the surface mesh, then its sector edges would induce a surface turn that is sharper than the precision of the EMST which directly corresponds to the precision of the 1-environment graph (cf. also Figure 4.4 of Chapter 4). This knowledge is also used to determine the order how the sectors are considered for insertion as shown in Section 8.3.

8.3 Sector Priority

In the second step, G is extended edge by edge, and suitable triangles identified during edge insertion are added to the initially empty triangulation M. For that purpose the sectors are processed according to some priority. For the following it is useful to introduce the definition of sector edges.

Definition 8.3 (Denotation of sector edges) *Let* w *be a sector of the cyclic arrangement of the edges incident to a vertex. Then*

$\mathbf{p}$ *denotes the vertex at which the sector is attached,*

$\mathbf{q}_1$ *and* $\mathbf{q}_2$ *denote the second endpoints of the two edges defining the sector,*

$e_1 := \overline{\mathbf{pq}_1}$, $e_2 := \overline{\mathbf{pq}_2}$,

$t(w)$ *denotes the triangle* $\triangle(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2)$ *induced by the sector.*

Algorithm 8.2 Computation of the optimal dihedral arrangement

Input: *Point* $\mathbf{p}$ *with incident edges and faces.*
Operation: *Calculation of an optimal dihedral arrangement:*
(1) $E :=$ *all current incident edges of* $\mathbf{p}$.
(2) $F :=$ *all current incident faces of* $\mathbf{p}$.
(3) Initialize $T(\mathbf{p}) := (E, F)$ *as the current triangulation at* $\mathbf{p}$.
(4) Compute the set $X \subseteq E$ *of edges with at most one incident face.*
(5) Compute the set W *of possible sectors* $w = (\overline{\mathbf{pq}_1}, \overline{\mathbf{pq}_2})$
with edges in X *so that* $\mathbf{q}_1$ *is in the 1-env. of* $\overline{\mathbf{pq}_2}$ *or* $\mathbf{q}_2$ *is in the 1-env. of* $\overline{\mathbf{pq}_1}$.
foreach (*sector* $w = (\overline{\mathbf{pq}_1}, \overline{\mathbf{pq}_2})$ *of* W) **do**
  Set t *as the induced face of* w.
  if (t *does not cause in* $T(\mathbf{p})$ *a dihedral angle smaller than* δ_c *(default:* $60°$*)*) **then**
    Add t *to* $T(\mathbf{p})$.
    Update W *according to the new adjacencies.*
  end
end
Compute the optimal dihedral arrangement according to Definition 8.1 with respect to $T(\mathbf{p})$.
Output: *The optimal dihedral arrangement for* $\mathbf{p}$.

As we already know, the triangle $t(w)$ should not have a too flat angle, and the dihedral angles with its adjacent triangles should not be too small. For these reasons we choose the sectors according to a value based on those two parameters.

Definition 8.4 *For the cyclic arrangement of the edges incident to a vertex* $\mathbf{p}$ *we define:*

$\mathrm{arctri}(w)$: *the angle between the edges* e_1 *and* e_2 *of a sector* w.

$\mathrm{arcdih}(e_i)$: *the dihedral angle between the two triangles incident to* e_i *which are induced by the two neighboring edges* e_i^+ *and* e_i^- *of* e_i *in the dihedral arrangement of* $\mathbf{p}$.

$\mathrm{arcdih}(w, \delta_c)$: *if* $\mathrm{arcdih}(e_1) \geq \delta_c$ *and* $\mathrm{arcdih}(e_2) \geq \delta_c$*, for the edges of* w*, then* $\mathrm{arcdih}(w, \delta_c)$ *is the average of* $\mathrm{arcdih}(e_1)$ *and* $\mathrm{arcdih}(e_2)$.
If $\mathrm{arcdih}(e_i) \geq \delta_c$ *for just one* i, $\mathrm{arcdih}(w, \delta_c) := \mathrm{arcdih}(e_i)$. *Otherwise,* $\mathrm{arcdih}(w, \delta_c) := 0$.

$\mathrm{b}(w)$: *if* w *is a non-1-environment sector then* $\mathrm{b}(w) := 1$*, and* $\mathrm{b}(w) := 0$ *else.*

The sectors w are processed in lexicographic order according to the key $(\mathrm{b}, \mathrm{arcdih}(w, \alpha), \mathrm{arctri}(w))$. The first and second components are processed in largest first order, and the third in smallest first order.

The reason for considering b is that the surface cannot curve along two adjacent edges which do not form an EMST for their three vertices, cf. the explanation of Chapter 4 and the related Figure 4.4.

The sorting order of the second key prefers sectors with flat environment. The intuition is that the favorable regions of the surface are preferably treated. This approach is in particular of advantage if the surface has sharp edges or ridges.

The priority queue Q stores the current candidate sectors. It is initialized with all sectors, and is updated by removing old and inserting new sectors emerging in the course of the algorithm if an edge is inserted into G. Thus the operations which Q has to support efficiently are access to the sector with the smallest key, insertion of an arbitrary sector, and deletion of an arbitrary sector.

In the following we describe the different cases (A to D) of Algorithm 8.1 in more detail.

8.4 Edge and Triangle Creation: Case A

If the angle of the triangle $t(w)$ at **p** is larger than a given global angle bound γ_c', $0° < \gamma_c' < 180°$, processing of the current sector is terminated without any further action. γ_c' is a parameter controlling the algorithm and is usually chosen large. That part of the algorithm is responsible for creating boundaries in the case that the desired surface is not closed. The mechanism of boundary generation caused by this part of the algorithm is described in Section 9.6. of Chapter 9.

8.5 Edge and Triangle Creation: Case B

Based on the current sector w, the algorithm tries to find an edge for insertion into G. Two main cases are distinguished by considering the triangle $t(w)$. If a point of P lies over $t(w)$, then the line s between **p** and **q** is inserted as an edge into G, where **q** is one of those points in the triangle for which s does not intersect any edge already in G. If no point of P is over $t(w)$, the algorithm continues with the else-case.

8.5.1 Candidate Points

Ideally, a point $\mathbf{p} \in P$ should be in the candidate region $C_c(t(w))$ if and only if **p** is in the NN-image of $t(w)$. Because the NN-image is not known, a selection strategy of candidate points is required which should yield a set of candidate points which comprehends the desired points. In the following we suggest a definition of a candidate region of triangles which has turned out to be sufficient in order to treat data sets successfully in practice. In Chapter 9 another definition of candidate region will be presented for which we will prove for SF-surfaces that it comprehends all points of P in the NN-image of the triangle, but also possibly some more.

The candidate region is defined as follows.

Definition 8.5 (Flat points over a sector) *Let w be a sector. A point **q** is called to be **flat over** w if the three largest dihedral angles in the tetrahedron $\tau = \diamond(t(w), \mathbf{q})$ are at the edges $\overline{\mathbf{qp}}$, $\overline{\mathbf{qq}_1}$, $\overline{\mathbf{qq}_2}$, that is, all angles are between triangles that are adjacent to **q**.*

*The **candidate region** $C_c^F(w)$ **of flat points over a sector** w is the region of all points in space which are flat over w.*

*Let P be a set of sample points of a surface. Then $P_c^F(w) := C_c^F(w) \cap P$ denotes the **candidate set of flat sample points over a triangle** w.*

The background of this definition is that triangles adjacent to $\mathbf{p} \in P_c^F(w)$ which satisfy the definition should fit well into the surrounding manifold because the dihedral angles should be large.

8.5.2 Point Selection

The required point in $P_c^F(w)$ is algorithmically determined by projecting $P_c^F(w)$ orthogonally onto $t(w)$. From the definition of $P_c^F(w)$ it is immediately clear that all projected points are in the interior of $t(w)$. A point **q** with the property is selected that a line in parallel to $\overline{\mathbf{q}_1\mathbf{q}_2}$ through **q** exists so that the triangle $\triangle(\mathbf{p}, \mathbf{q}_1', \mathbf{q}_2')$ cut off by this line from $t(w)$ does not contain any of the projected points in its interior. Evidently, such a line exists. If $\overline{\mathbf{pq}}$ does not χ-intersect any edge of G then **q** is taken as the desired point. If the unlikely case of an intersection happens then the overall algorithm continues with the next sector.

8.5.3 Edge Insertion

The main task of the insertion procedure for the candidate edge $e := \overline{\mathbf{pq}}$ is to find locations for e in the cyclic arrangements of **p** and **q**. For that purpose, an optimal dihedral arrangement is recalculated at **p** and **q** under consideration of the new edge e, using Algorithm 8.2. This procedure yields the location of e in the arrangements.

Furthermore, e is tested for existence of an intersection with a triangle in M. The intersection test with the triangle is performed in order to keep M definitively intersection-free, although the probability that e intersects a triangle should be small for properly sampled surfaces. If no intersection is found then e is inserted into M.

For both vertices **p** and **q**, the sectors which have been destroyed by this operation are removed from the priority queue Q, and the newly created sectors are inserted, according to their keys.

8.6 Edge and Triangle Creation: Case C

If e_3 is an edge of G then the triangle $t(w)$ is a candidate for insertion into M. $t(w)$ is tested for whether it intersects an edge of G, in order to guarantee intersection-freeness of the resulting manifold. For properly sampled surfaces this case should not happen but this test additionally enforces that M is intersection-free. If $t(w)$ is intersection-free then it is inserted into M.

If e_3 is not an edge of G then the algorithm continues with the then-case. The graph G and the priority queue Q remain unchanged.

8.7 Edge and Triangle Creation: Then-Case D

One purpose of case D is to extend the current triangulation in a ”non-convex” manner by including further points not yet covered by the ”hull” of the already existing triangulation, as it is done for the points that are flat over a triangle. This strategy is crucial for the algorithm. The idea is to check at a possible new boundary edge e_3 given by an edge closing a sector to a triangle $t(w)$ whether the boundary can be extended instead of being ”closed”. The candidate sample points are found in the candidate environment of e_3.

Another purpose is to avoid triangles with large angles from which we know from Chapter 6 that they are unfavorable for NN-embeddability. A large angle may occur if an edge of type e_3 is inserted although a vertex **q** is close to it. In this case the edge $\overline{\mathbf{pq}}$ usually is a better candidate.

8.7.1 Candidate Region of a Line Segment

The definition of a candidate region of a line segment used by the algorithm is based on the β-environment of Definition 5.8. Its definition is preceded by a further definition required for its formulation.

Definition 8.6 (**β-close point**) *Let s be a line segment and* **p** *be a point.* **p** *is called **β-close to** s if* **p** *is in the β-environment of s (cf. Definition 5.8).*

*If s does not have β-close points then s is called **β-line segment**.*

Definition 8.7 (**β_c-candidate region of a sector**) *Let w be a sector. A point* **q** *is called to be **β_c-before** w if*

(1) $\mathbf{q}$ *is* β_c*-close to* e_3*,*

(2) $\mathbf{q}$ *projects orthogonally outside of* $t(w)$*, on the opposite side of* $\mathbf{p}$ *with respect to the line through* e_3.

The β_c***-candidate region*** $C^{\beta_c}_{c,+}(w)$ ***before a sector*** w *is the region of all points in space which are* β_c*-before* w*.*
Let P *be a set of sample points of a surface. Then* $P^{\beta_c}_{c,+}(s) := C^{\beta_c}_{c,+}(w) \cap P$ *denotes the* β_c***-candidate set before*** w*.*
A point $\mathbf{q}$ *is called to be* β_c***-close over*** w *if*

(1) $\mathbf{q}$ *is* β_c*-close to* e_3*,*

(2) $\mathbf{q}$ *projects orthogonally into* $t(w)$*, and*

(3) $\mathbf{q}$ *is not flat over* w*.*

The ***candidate region*** $C^{\beta_c}_{c,-}(w)$ ***of*** β_c***-close points over a sector*** w *is the region of all points in space which are* β_c*-close over* w*.*
Let P *be a set of sample points of a surface. Then* $P^{\beta_c}_{c,-}(w) := C^{\beta_c}_{c,-}(w) \cap P$ *denotes the* ***candidate set of*** β_c***-close points over*** w*.*
The β_c***-candidate region of a sector*** w *is defined by* $C^{\beta_c}_c := C^{\beta_c}_{c,+}(w) \cup C^{\beta_c}_{c,-}(w)$*, and the* β_c***-candidate set of a sector*** w *by* $P^{\beta_c}_c(w) := P^{\beta_c}_{c,+}(w) \cup P^{\beta_c}_{c,-}(w)$*.*

The index c of β_c indicates that β_c is one of the control parameters of the algorithm. The β_c used for the candidate environment can be that one of the clustered β-environment graph of the first phase of the reconstrucion algorithm. As we will see later on, smaller values of β_c, including negative ones, also may be used if the sample set is suitable chosen. Later examples will show that $\beta_c = -0.5$ is a suitable value for surfaces without sharp edges or ridges.

8.7.2 Point Selection

The candidate sets $P^{\beta_c}_c(s)$, $P^{\beta_c}_{c,+}(s)$, and $P^{\beta_c}_{c,-}(s)$ are further restricted by the condition of step D to those points which do not cause a fold-over quadrilateral.

Definition 8.8 (Fold-over quadrilateral) *Let* w *be a sector, and* $\mathbf{q}$ *be a vertex. The quadrilateral* $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ *has a* ***fold-over*** *if the dihedral angle between the triangles* $\triangle(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2)$ *and* $\triangle(\mathbf{p}, \mathbf{q}_1, \mathbf{q})$ *or* $\triangle(\mathbf{p}, \mathbf{q}_2, \mathbf{q})$ *is larger than the dihedral angle between the triangles* $\triangle(\mathbf{p}, \mathbf{q}_1, \mathbf{q})$ *and* $\triangle(\mathbf{p}, \mathbf{q}_2, \mathbf{q})$*.*

Definition 8.9 (Fold-over-free candidate sets) *The* ***fold-over-free candidate sets*** $\overline{P}^{\beta_c}_c(s)$*,* $\overline{P}^{\beta_c}_{c,+}(s)$*, and* $\overline{P}^{\beta_c}_{c,-}(s)$ *are defined as the restrictions of the candidate sets* $P^{\beta_c}_c(s)$*,* $P^{\beta_c}_{c,+}(s)$*, and* $P^{\beta_c}_{c,-}(s)$ *to those points which do not cause a fold-over quadrilateral.*

The point selection procedure consists of two search steps. The second search step depends on the result of the first step.

Search step 1:

A proper point of the candidate set $\overline{P}_c^{\beta_c}$ is selected as follows. From all points of $\overline{P}_c^{\beta_c}$ the one enclosing the largest angle with $\overline{\mathbf{q}_1\mathbf{q}_2}$ is taken. If the 1-environment $E_1(\mathbf{p},\mathbf{q})$ is not empty of points of $\overline{P}_{c,-}^{\beta_c}(w)$, then $\mathbf{q}$ is replaced by the point of $\overline{P}_{c,-}^{\beta_c}(w) \cap E_1(\mathbf{p},\mathbf{q})$ nearest to $\mathbf{p}$. This process is repeated for the new $\mathbf{q}$ with the edge $\overline{\mathbf{pq}}$ until no further update is possible. Because the distance of the investigated points from $\mathbf{p}$ as elements of the 1-environment of their predecessor decreases strictly, termination is guaranteed.

Search step 2, inside case:

If the projection of $\mathbf{q}$ falls inside of $t(w)$, the line segment $\overline{\mathbf{pq}}$ is checked for χ_c-intersection with an already existing graph edge. If an intersection is found then the points in $\overline{P}_{c,-}^{\beta_c}$ which are closer to $\mathbf{p}$ than $\mathbf{q}$ are processed in order of increasing distance from $\mathbf{p}$. If a point $\mathbf{q}'$ is found for which $\overline{\mathbf{pq}'}$ does not χ_c-intersect an already existing graph edge then $\mathbf{q}'$ is taken for $\mathbf{q}$. Otherwise the selection of a suitable $\mathbf{q}$ fails and the else-case-D is executed.

Search step 2, outside case:

If the projection of $\mathbf{q}$ falls outside of $t(w)$, then the edges of type $\overline{\mathbf{q}_1\mathbf{r}_1}$ or $\overline{\mathbf{q}_2\mathbf{r}_2}$ already incident to $\mathbf{q}_1$ and $\mathbf{q}_2$, respectively, are investigated. Among them, one is taken which encloses the smallest angle at $\mathbf{q}_1$ ($\mathbf{q}_2$) with the edge $\overline{\mathbf{q}_1\mathbf{q}_2}$ and for which the orthogonal projection of $\mathbf{r}_1$ ($\mathbf{r}_2$) onto the plane of $t(w)$ falls onto the opposite side of $\mathbf{p}$ with respect to the line through $\mathbf{q}_1$ and $\mathbf{q}_2$. If the angle of the chosen edge with $\overline{\mathbf{q}_1\mathbf{q}_2}$ at $\mathbf{q}_1$ ($\mathbf{q}_2$) is less than that of $\overline{\mathbf{q}_1\mathbf{q}}$ ($\overline{\mathbf{q}_2\mathbf{q}}$), then the point $\mathbf{q}$ is updated by that one of $\mathbf{r}_1$ and $\mathbf{r}_2$ which yields the larger of the angles $\mathrm{arc}(\overline{\mathbf{r}_1\mathbf{q}_1}, \overline{\mathbf{r}_1\mathbf{q}_2})$ and $\mathrm{arc}(\overline{\mathbf{r}_2\mathbf{q}_1}, \overline{\mathbf{r}_2\mathbf{q}_2})$.

If $\mathbf{q}$ has been modified in the final search and if a point of the set $\overline{P}_c^{\beta_c} - \{\mathbf{q}', \mathbf{r}_1, \mathbf{r}_2\}$ is in the 1-environment of $\overline{\mathbf{pq}}$ or in the double cone which results by rotation of the edges $\overline{\mathbf{qq}_1}$,$\overline{\mathbf{qq}_2}$ around the edge $\overline{\mathbf{q}_1\mathbf{q}_2}$, then the search for a proper $\mathbf{q}$ has not been successful, where $\mathbf{q}'$ is the previous "old" unmodified $\mathbf{q}$. The same holds if the new point $\mathbf{q}$ is not an element of the candidate set $\overline{P}_c^{\beta_c}$. If this happens then the else-case D is executed.

If a suitable $\mathbf{q}$ has been found then the quadrilateral induced by $\mathbf{p}$,$\mathbf{q}$,$\mathbf{q}_1$,$\mathbf{q}_2$ has to be triangulated. The details are described in the next subsection.

The background of the described selection strategy is as follows. The goal is to extend the set of vertices adjacent to $\mathbf{p}$ by a vertex of the candidate set $\overline{P}_c^{\beta_c}$. The evidently best suited point is the one which is "closest" to edge e_3. However, if a sharp edge occurs close to the sector, it can be useful to consider further points, too. Let us look at Figure 8.3. The figure shows a side view of a sector in which $\mathbf{q}_1$ and $\mathbf{q}_2$ fall onto one another. $\mathbf{q}'$ is in the 1-environment of $\overline{\mathbf{pq}}$. The selection of $\overline{\mathbf{pq}}$ would yield a surface which goes from $\mathbf{p}$ over $\mathbf{q}$ to $\mathbf{q}'$. This surface, however, has a higher curvature than a surface from $\mathbf{p}$ over $\mathbf{q}'$ to $\mathbf{q}$. The search step 1 takes this observation into account and selects $\mathbf{q}'$ in this situation.

The search step 2, inside case, is executed in order to avoid intersecting edges. In our empirical investigations this case never occurred up to now.

The search step 2, outside case, takes into account the current mesh structure "in front" of the triangle. If adjacent points $\mathbf{r}_1, \mathbf{r}_2$ at $\mathbf{q}_1, \mathbf{q}_2$ are already present, it might be that they induce triangles $\triangle(\mathbf{q}_1, \mathbf{q}_2, \mathbf{r}_1)$, $\triangle(\mathbf{q}_1, \mathbf{q}_2, \mathbf{r}_2)$ with the edge $\overline{\mathbf{q}_1\mathbf{q}_2}$ for which the current candidate point $\mathbf{q}$ "is behind them" with respect to the edges $\overline{\mathbf{q}_1\mathbf{r}_1}$, $\overline{\mathbf{q}_2\mathbf{r}_2}$. In Figure 8.4 (left), the point $\mathbf{q}$ "is behind" the triangle $\triangle(\mathbf{q}_1, \mathbf{q}_2, \mathbf{r}_2)$ because it induces a larger angle at $\mathbf{q}_2$ with $\overline{\mathbf{q}_1\mathbf{q}_2}$ than $\mathbf{r}_2$ does. However, insertion of the edge $e_3 = \overline{\mathbf{q}_1\mathbf{q}_2}$ would induce a relatively small dihedral angle at $\overline{\mathbf{q}_1\mathbf{q}_2}$ between the triangles

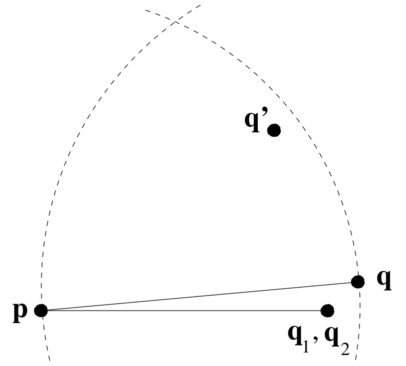


Figure 8.3: The side view of a configuration in which $\mathbf{q}'$ is selected instead of $\mathbf{q}$.

$\triangle(\mathbf{q}_1, \mathbf{q}_2, \mathbf{r}_2)$ and $\triangle(\mathbf{q}_1, \mathbf{q}_2, \mathbf{p})$, cf. Figure 8.4 (right). Therefore it makes sense to consider the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2, \mathbf{r}_2)$ to determine the correct triangulation. In order to do this $\mathbf{r}_2$ replaces $\mathbf{q}$ as the "new" $\mathbf{q}$.

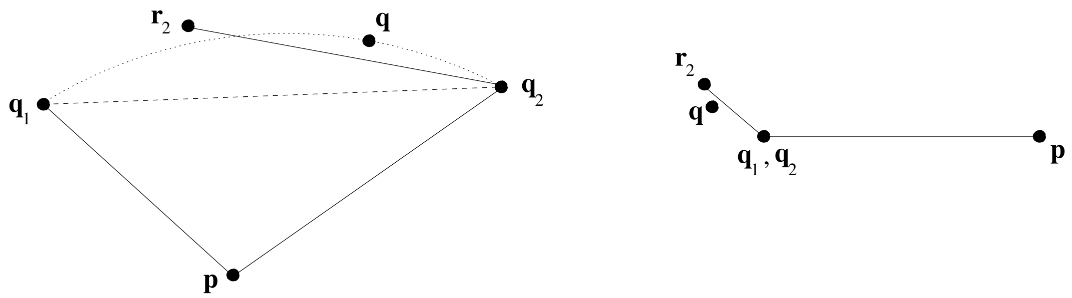


Figure 8.4: The point $\mathbf{q}$ "is behind" the edge $\overline{\mathbf{q}_2\mathbf{r}_2}$ (left). The side view (right) shows that it makes sense to consider $\mathbf{r}_2$ as new candidate point because the other triangulation of the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2, \mathbf{r}_2)$ with the edge $\overline{\mathbf{p}\mathbf{r}_2}$ could yield a smaller dihedral angle.

8.7.3 Triangulation

The four points $\mathbf{p}$,$\mathbf{q}_1$,$\mathbf{q}_2$,$\mathbf{q}$ are triangulated according to the min-max triangulation, that is the maximum angle of the resulting triangles is minimized [HL92].

An alternative approach might be to take that triangulation for which the dihedral angle between the resulting triangles is greater, and, in the case of equality (which happens in particular if the four points are co-planar), the min-max triangulation. We did try this possibility, too, and did not recognize a significant different behavior.

Another alternative is to take the environment of the four vertices in the current surface mesh and graph into account, with the goal to keep the surrounding mesh smooth. The idea is to maximize the occurring dihedral angles. The edges of the two possible triangulations of $\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2$ are virtually inserted into the current surface mesh, and the dihedral angles of the two resulting meshes are estimated by taking the current dihedral angles at the five edges in the optimal dihedral arrangement of each of the four vertices. From those of the resulting 10 values which exceed δ_c the average is taken. Then the triangulation of $\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}$ with the greater average is chosen, and that line segment among $\overline{\mathbf{pq}}$ and $\overline{\mathbf{q}_1\mathbf{q}_2}$ is chosen as candidate edge for insertion which belongs to the selected triangulation.

Figure 8.5 shows two examples for triangulation by min-max edge selection and edge selection under consideration of the environment. The second version follows the surface more accurately.

It may also make sense to combine the two versions. In Chapter 9 we will see that it is important that the algorithm avoids edges for which sample points exist which form a large angle with them. Thus, if **q** forms a large angle with e_3, $\overline{\mathbf{q}_1\mathbf{q}_2}$ should become the candidate edge which is best achieved by applying the version based on min-max triangulation. Otherwise priority can be given to the dihedral smoothness of the mesh by applying the version which takes the neighborhood into consideration.

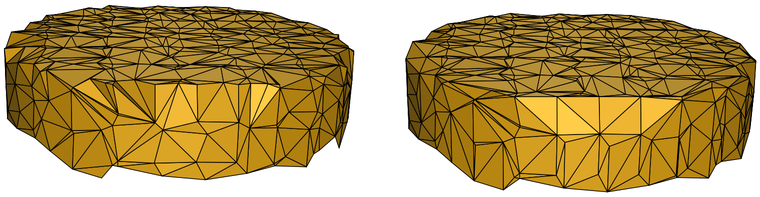


Figure 8.5: Left: a triangulation obtained with the min-max edge selection. Right: a triangulation obtained by taking the environment into account. The second version yields an improved reconstruction at the sharp edge.

8.7.4 Edge Insertion

The rest of the insertion procedure of a candidate edge of type $e = \overline{\mathbf{pq}}$ is the same as in Section 8.5.3. For a candidate edge e_3 the insertion procedure has to find proper locations for the new edge in the cyclic edge arrangements of its vertices $\mathbf{q}_1$ and $\mathbf{q}_2$. This is achieved by applying Algorithm 8.2. The results yield the location of e_3 in the arrangements.

Furthermore, e_3 is tested for existence of an intersection with a triangle in M. The intersection test with the triangle is performed in order to keep M definitively intersection-free, although the probability that e_3 intersects a triangle should be small for properly sampled surfaces. If no intersection is found then e_3 is inserted into M.

For both vertices **p** and **q**, the sectors which have been destroyed by this operation are removed from the priority queue Q, and the newly created sectors are inserted, according to their keys.

If e_3 could be inserted into G then the insertion procedure for the triangle $t(w)$ is executed which works as described in Section 8.6.

8.8 Edge and Triangle Creation: Else-Case D

In the else-case D, the edge e_3 has to be checked for intersection with an already existing graph edge using the χ_c-intersection test. It is unlikely that an intersection is really found. If no intersection has been determined, e_3 is inserted with the same procedure as described in Section 8.7.4. If e_3 could be inserted into G then the insertion procedure for the triangle $t(w)$ is executed which works as described in Section 8.6.

8.9 Examples

Figures 8.6 and 8.7 show the results of reconstruction of our example data sets. The 1-environment graph has been used as surface description graph. Several reconstructed surfaces have boundaries,

object	# points	computation times in minutes and seconds		
		DT(P)	clustered 1-EG	reconstruction
torus	310	0:01	0:01	0:03
cup	2650	0:11	0:06	0:25
head	1487	0:06	0:03	0:14
skull	698	0:03	0:01	0:05
puppet	695	0:03	0:01	0:07
cap	371	0:02	0:01	0:02
pharaoh	2286	0:10	0:05	0:20
tori	620	0:01	0:02	0:06

Table 8.1: The calculation times of the reconstruction algorithm for the examples on an SGI Octane R10000 at 250 MHz with 384 MByte of memory.

and the skull additionally has holes which have been properly reconstructed. The parameters used are listed in the caption of the figures. The calculation times are compiled in Table 8.1. The Delaunay triangulation $DT(P)$ has been used as basic data structure for the nearest neighbor queries during the computation (see Appendix B).
Figure 8.8 shows a sequence of snapshots of the triangulation phase for the clustered β-environment graph with $\beta = 1$ as initial surface description graph for the triangulation.

8.10 Computational Issues

The implementation of the basic data structures of the reconstruction system is based on a *library of template classes*. The template classes offer data structures which are used at different locations of the implementation. In the following section we give a brief survey on these classes. Then the data structure of the partially ordered surface description graph is described. That data structure supports the combinatorial requirements of the algorithm, as does the priority queue of sectors to which a subsection is devoted, too. Finally some aspects of geometric processing, required in particular for the edge feasibility tests are treated.

8.10.1 A Library of Template Classes

Sorting and containment tests can be carried out with an *AVL tree template class*. The implementation of a *set class* for simple set arithmetics is also based on AVL trees [Wir86].
Whenever sorting is required where the number of objects is not known and where only *the first k elements* of a set have to be extracted the *heap template class* can be chosen. As AVL trees, it has a run time proportional to $\mathcal{O}(n \log n)$, but for sorting only subsets of elements *heaps* are the better choice.
Of course, *template arrays* are also part of the library as well as simple *list templates*, *queue templates*, *stack templates*, and *standard string classes*.
One interesting part of the library is the integration of so-called *hash list classes*. Here, the number of elements to be inserted and the *hash function* can be adjusted by the programmer and the system automatically generates an appropriate *hash table size*. Then, elements can be inserted in this list or the position of an element can be extracted in constant time.
Space subdivisions for *intersection tests*, for example, can be realized using several implemented *space subdivision template classes*. The basic idea common to all schemes is to subdivide a bounded space

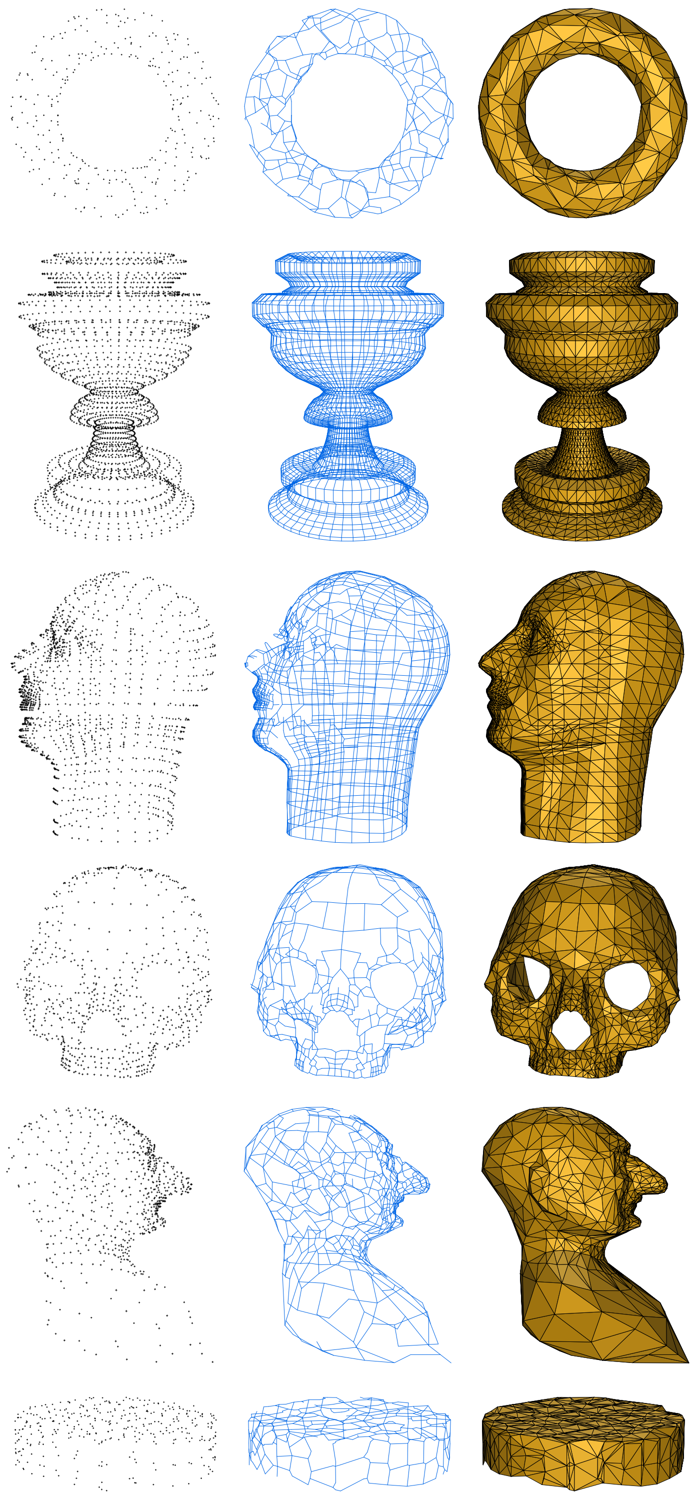


Figure 8.6: Results of the reconstruction algorithm. From left to right: the point set, the clustered 1-environment graph and the reconstruction result. Parameter values: $\gamma_c' = 135°$, $\beta_c = 1$, $\chi_c = 75°$, $\delta_c = 60°$. An exception is the skull. In order to reconstruct even the hole of the nose and not just the eyeholes, a boundary detection value of $\gamma_c' = 120°$ has been used.

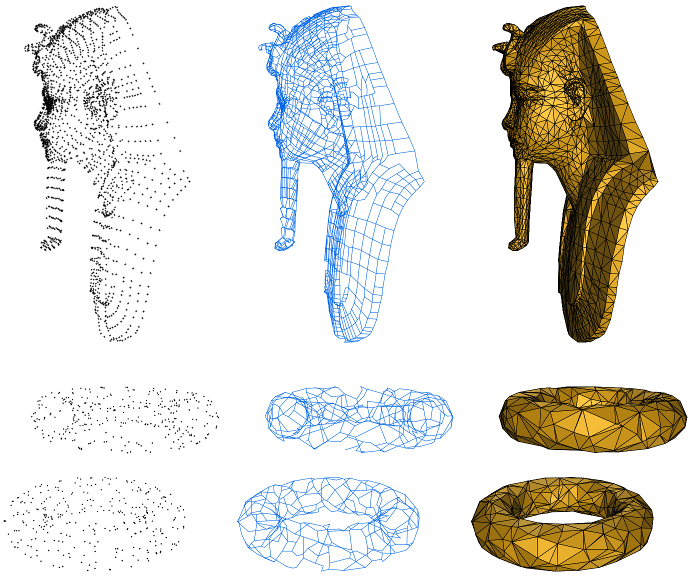


Figure 8.7: Results of the reconstruction algorithm. From left to right: the point set, the clustered 1-environment graph and the reconstruction result. The parameter values were $\gamma_c' = 135°$, $\beta_c = 1$, $\chi_c = 75°$, $\delta_c = 60°$.

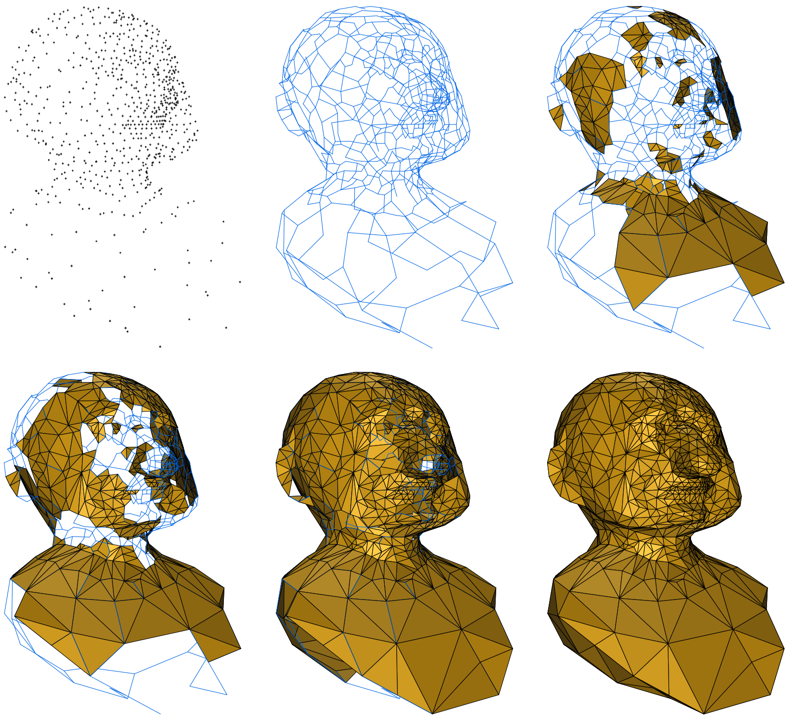


Figure 8.8: Snapshots of the process of triangulation of the puppet. From left to right: the point set, the clustered 1-environment graph, and the intermediate reconstruction results with 300, 600, 1100 triangles, and the final reconstruction result.

filled with geometric objects recursively into sub-cells. To every leaf cell, its intersecting objects are assigned. This data structure can be used to answer incidences of a query object with the given geometric object. The query is processed by traversing the hierarchy for those leaves which intersect the query object. Then the incidence test is executed with the objects of the leaves found.

Our library contains an *octree template class* and a *median cut tree template class* working according to that scheme. Other subdivision schemes can be derived very easily from the base implementation. Using these classes the programmer must only provide a so-called *find function*, which specifies for each query in the tree whether it was successful or not. Such a *find function* can for instance be a simple *intersection test*.

8.10.2 Data Structure of the Partially Embedded Surface Description Graph

The data structure of the partially embedded surface description graph has to support the following basic operations:

- initialization,
- output of the coordinates of a vertex,
- location of an existing edge in a cycle of edges of a vertex,
- ordered edge insertion behind or before a given edge in the cycle of edges of a vertex,
- re-arrangement of the cycle of edges incident to a vertex,
- output of the cycle of edges incident to a vertex,
- next/preceding edge on the cycle of edges of a vertex,
- surface triangle insertion,
- incident surface triangles of an edge,
- output of all surface triangles.

As we can see from this list of operations, the data structure also contains the resulting surface triangles, in addition to the usual graph information. That in particular means that we do not need a separate data structure for the resulting manifold. Its triangles can be accessed by the operation ”output of all surface triangles”.

For the data structure of the partially embedded surface description graph we use the incidence list representation as basic framework. The incidence list representation consists of a list of vertex elements. Every vertex element refers to a list of edge elements. The elements of the edge list of a vertex are arranged in sorted order of the edge cycle of the vertex. They refer to the two vertex list elements of their two vertices.

The surface triangles are stored in another list. Every element of the edge lists refers to at most two triangles that are incident to the edge. A triangle element refers to the edge list elements of its three edges.

Every vertex, edge, and triangle element stores an identifier which uniquely describes the represented item.

8.10.3 Priority Queue of Sectors

The algorithm processes relevant sectors in sorted order according to a key which has been described earlier. The data structure of the set of sectors still to be processed has to support the following operations:

- get and remove the first element in sorted order according to its key,
- insert a sector according to its key,
- remove an arbitrary sector specified by an identifier.

The first two operations are those of a classical priority queue and any of the well-known data structures is used. The last operation is required if an update of the surface description graph by a new edge is performed. The access required for that operation is implemented by a pointer referring from the element of the first edge of a sector in cyclic order to its sector element in the priority queue.

8.10.4 Geometric Tests

The geometric tests to be performed and the corresponding computational solutions are as follows.

8.10.4.1 χ-intersecting edges

The task is to find edges of the surface description graph which are close to a given edge e. Closeness of two edges is measured by considering dihedral angles between certain triangles of the tetrahedron spanned by the four vertices of the two edges. In order to restrict search space, we consider a ball around the center of every edge. The radii of the balls are at least half of the edge length. That means that the vertices of the edge are within the closed ball. For a given edge e, we only consider those graph edges e' which do not have both of the following properties:

(1) The ball of e' does not contain any of the vertices of e.

(2) The ball of e does not contain any of the vertices of e'.

The background of this discussion is the definition of conflict-freeness of Section 7.2. A pair of edges satisfying (1) and (2) is conflict-free in that sense. Furthermore, if r is half of the length of the longer one of the two edges, and we take $r_B = c \cdot r$ as ball radius, then the distance of the two edges is at least $r_B - 3r = (c - 3) \cdot r$. The reason is that the edges are in the same balls of radius r, but the large balls of radius r_B do not completely cover the small balls because otherwise (1) or (2) is satisfied. If $c > 3$ is chosen sufficiently large, the dihedral angles between the triangles under consideration are far from $180°$ if (1) and (2) hold.

In order to test the negation of (1) efficiently, for every vertex $\mathbf{v}$ of the graph the set of all edges is stored whose ball contains $\mathbf{v}$. For an edge, these vertices can be determined by k-nearest-neighbor search. Then for the query edge e, the candidate edges are immediately obtained from the lists of the vertices of e.

The edges e' not satisfying (2) are found by a k-nearest-neighbor search around the center of e which is terminated if the radius of the ball of e is reached. For k-nearest-neighbor search, the approach of Section B.1 of Appendix B is used.

If the edges are short, as can be expected in our application, (1) and (2) yield a significant restriction of search space.

8.10.4.2 Points flat over a triangle

The main task is to restrict the set of points for which the criterion of flatness over a triangle has to be evaluated. This problem is solved by using the k-nearest-neighbor search for the point $\mathbf{p}$ of a sector w, so that the search space covers the NN-image of the triangle. An estimation of this search space is discussed in Section 9.5.2 of Chapter 9.

8.10.4.3 β-close points of an edge

The task is to find the vertices $\mathbf{p}'$ possibly located in an $E_\beta(e)$-environment of a given edge e. This problem also has occurred for the calculation of β-environment graphs, and is solved as outlined in Section 5.3 of Chapter 5.

8.10.4.4 Intersecting triangles

The task is to find out for a given triangle between graph vertices whether it intersects an edge of the graph, and for a line segment between two graph vertices whether it intersects a triangle of a given set of triangles between graph vertices. These problems are solved by using one of the spatial subdivision schemes for answering incidence queries of our template library, outlined in Section 8.10.1. Another possibility for the intersection test is to apply a similar approach as for χ-intersecting edges in Section 8.10.4.1 in order to reduce the search space. However, if a line segment intersects a triangle it usually also intersects an edge of the triangle so that this task is usually covered by the edge intersection tests.

8.11 Discussion

The algorithm of triangulation presented in this chapter is the second phase of the reconstruction algorithm. The computational examples have shown that it works well. In the next chapter, arguments will be given which show that the favorable behavior is not surprising. Chapter 10 will show that the triangulation algorithm can also be applied usefully to graphs different from the type of surface description graphs used in the first phase of the algorithm.

Chapter 9

Analysis of Triangulation

In the following we investigate the usefulness of the triangulation algorithm. We first introduce a characterization of suitable sample sets and prove the existence of sample sets satisfying this condition. Then we show for the flat case that sample sets fulfilling this condition exist for which the probability that

(1) triangles with inner angles larger than γ occur,

(2) edges of the triangulation become longer than a given bound,

is rather low. "Probability" means the portion of all configurations for which an assertion holds. This means that we assume that every configuration may occur with equal probability.
Afterwards we argue that the same holds in space, too. Finally, by applying theorems of Chapter 6, we can show that the probability that the algorithm yields a reconstruction in the sense of the mentioned chapter should be high.
Details of the theoretical analysis are confirmed by empirical investigations on random sample sets.

9.1 Characterization of Sample Sets Suitable for Triangulation

According to the results of Chapter 6 on the NN-embeddability of triangulations, a sample set is considered suitable if the triangulation algorithm does not yield long edges, and if the angles of the generated triangles do not become too large. In the following we combine both aspects in order to characterize a class of favorable sample sets. For that purpose, the following terminology is useful.

Definition 9.1 (γ-line segment and γ-edge) *A point* $\mathbf{r}$ *is* γ***-close*** *to a line segment* $s = \overline{\mathbf{r}_1\mathbf{r}_2}$ *if the angle of triangle* $\triangle(\mathbf{r}_1, \mathbf{r}, \mathbf{r}_2)$ *at vertex* $\mathbf{r}$ *is larger than* γ*,* $90° \leq \gamma < 180°$*.*
An edge of a graph is called a γ***-edge*** *if it does not have a* γ*-close vertex in the graph. A line segment between two vertices of the graph is called a* γ***-line segment*** *if it does not have a* γ*-close vertex in the graph.*

This terminology is closely related to the terminology which has been used for β-environments, cf. e.g. Definition 5.8. This is not surprising because we know from Theorem 5.11 that the vertices γ-close to s are just those which are $\beta(\gamma)$-close to s with $\beta(\gamma) := (\cot\gamma)/2$, $90° \leq \gamma < 180°$. Vice-versa, for $\beta \leq 0$, the vertices β-close to s are just those which are $\gamma(\beta)$-close to s for $\gamma(\beta) := \arcsin(\frac{1}{\sqrt{1+4\beta^2}})$, $90° \leq \gamma(\beta) < 180°$.
The idea behind the following characterization of suitable sample sets is as follows. If for long line segments between points of the sample set γ-close sample points would exist, and if the algorithm

would yield a triangulation which does not contain triangles with angles larger than γ, then the algorithm would not yield long edges. Because the triangulation algorithm is designed to avoid triangles with large angles, long edges should be avoided by it, too.

For the initial SDG no special constraints have to be defined for the sample sets.

Theorem 9.2 (γ-edge property of β-graphs) *The edges of a β-EG, $0 \leq \beta \leq 1$, and thus the edges of the SDG resulting from step 1 of the reconstruction algorithm, are γ-edges for $90° \leq \gamma < 180°$.*

Proof: From Theorem 5.11 we know that the vertices γ-close to s are just those in the β-environment of s with $\beta(\gamma) := (\cot \gamma)/2$, $90° \leq \gamma < 180°$. Because the $\beta(\gamma)$-environments for $90° \leq \gamma < 180°$ are subsets of the β-environments for $0 \leq \beta \leq 1$, and the β-environment of β-EG-edges is free of points, the edges of the β-EG do not have γ-close points. ∎

For the subsequent triangulation, we proceed as follows.

Theorem 9.3 (β-blocking sampling)
Let S be a compact SF-surface without boundary, $-\infty < \beta \leq 1$.

(1) An $l'_0 > 0$ exists so that every line segment $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in S$, $l(s) \leq l'_0$, has a β-close point on S.

(2) Let be $0 < l_0 < l'_0$, P a finite set of points on S. Then a finite extension P' of P by points on S exists so that for every line segment $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in S$, $l_0 \leq l(s) \leq l'_0$, a point $\mathbf{r}'$ in P' exists which is β-close to s. Every further extension of P' by points on S has this property, too.

Proof:

(1) Let be $\mathbf{m} := \frac{1}{2}(\mathbf{p} + \mathbf{q})$, and $E_g(\mathbf{m})$ be the open ball with maximum radius $g(s)$ and center $\mathbf{m}$ which is a subset of the β-environment of s. By Theorem 7.19 and Theorem 6.4, $l'_0 > 0$ exists so that the maximum distance of every point of s, and thus of $\mathbf{m}$, to S is less than $g(s)$ if $l(s) \leq l'_0$. Thus the nearest neighbor $\mathbf{r} \in S$ of $\mathbf{m}$ satisfies the requirements of (1).

(2) Let L be the set of all line segments s between points of S with $l_0 \leq l(s) \leq l'_0$. Because S is compact, L is compact.

For points $\mathbf{r} \in S$, let $E(\mathbf{r})$ be the set of all $s \in L$ which contain $\mathbf{r}$ in their open β-environment. $E(\mathbf{r})$ contains an open subset. The reason is that the function which maps s to the angle of $\mathbf{r}$ in the triangle $\triangle(\mathbf{p}, \mathbf{r}, \mathbf{q})$ is continuous. By (1), the sets $E(\mathbf{r})$, $\mathbf{r} \in S$, cover L. By the finite-covering theorem of topology, a finite set P'' exists so that the sets $E(\mathbf{r})$, $\mathbf{r} \in P$, cover L, too. $P' := P \cup P''$ has the property desired in (2). Trivially, every further extension does have this property, too.

∎

Definition 9.4 (Samp_2-property) *Let be $0 < l_0 < l'_0$, and S be a surface. A finite set P of points on S has the $\mathrm{Samp}_2(l_0, l'_0)$**-property** if for every line segment $s = \overline{\mathbf{pq}}$, $\mathbf{p}, \mathbf{q} \in S$, $l_0 \leq l(s) \leq l'_0$, a point $\mathbf{r}$ in P exists which is β-close to s.*

Corollary 9.5 (Existence of Samp_2-sample sets) *Let S be a compact SF-surface without boundary, $-\infty < \beta \leq 1$, and P be a finite set of points in S. Then l'_0 and a finite extension P' of P by points on S exist so that P' has the $\mathrm{Samp}_2(l_0, l'_0)$ property for every l_0 with $0 < l_0 < l'_0$.*

Proof: The corollary is an immediate implication of Theorem 9.3. ∎

In the following we formalize the idea concerning the edge length bounds outlined at the beginning of the section.

Observation 9.6 (Existence of edge length bounding sample sets) *Let S be a compact SF-surface without boundary, $90° \leq \gamma < 180°$, P a finite set of points on S. Then $l_0'' > 0$ exists so that for all l_0 with $0 < l_0 < l_0''$, P can be augmented by points on S to a finite sample set P' with the property that the graphs G constructed by the algorithm should have edge length less than l_0, as long as the triangulation algorithm just inserts γ-edges.*

Argumentation: By Corollary 9.5, l_0' and a finite extension P_1' of P by points on S exist so that P' has the $\mathrm{Samp}_2(l_0, l_0')$ property for $\beta(\gamma)$, for every l_0 with $0 < l_0 \leq l_0'$, and so does every finite superset P'.

By Observation 7.10, $l_{\min} > 0$ exists so that, for all l_0, $0 < l_0 < l_{\min}$, P can be extended to a finite point set P_2' for which the clustered β-environment graph, $0 \leq \beta \leq 1$, of P' should only have edges of length less than l_0, and so does every finite superset P'.

Then $l_0'' := \min\{l_0', l_{\min}\}$ and $P' := P_1' \cup P_2'$ fulfill the requirements of the observation. ■

The implication of this observation is that all theorems which demand short edges in order to achieve a property can be applied to G and M, as long as the triangulation algorithm just inserts γ-edges.

9.2 General Edge Length Bounds

In Section 9.1 we have seen that sample sets exist by which a given edge length bound is achieved. In this section we investigate the behavior of edge lengths for the case of arbitrary, unconstraint sample sets, in order to see to what extent proper sampling is indeed necessary. It turns out that in many cases the length of a newly inserted edge is not longer than edges already in the graph.

For inserted edges of type e_3, not too much special can be told. In the worst case, the length of e_3 can come close to the sum of the lengths of e_1 and e_2 where the minimum possible difference depends on the choice of γ_c'. The initial SDG in form of a clustered β-environment graph, $0 \leq \beta \leq 1$, may contain triangles, although for $\beta = 1$ the probability of triangles is low (Theorem 5.6). The worst case of length for e_3 occurs for $\beta = 0$ where $l(e_3) \leq \sqrt{l(e_1)^2 + l(e_2)^2}$. The reason is that in the worst case $\mathbf{p}$ is on the diameter sphere of e_3 so that the triangle $\triangle(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2)$ is rectangular. A favorable situation may occur if e_3 is the edge inserted by the min-max triangulation of the then-case of D.

The following theorem gives an estimation of the length of edges of type $\overline{\mathbf{pq}}$ dependent on the shape of a sector w. These estimations are independent from the condition that G does only contain γ-edges.

Theorem 9.7 (Length bound for $\overline{\mathbf{pq}}$)
Let be

- *w a sector,*
- *H the plane spanned by $t(w)$,*
- *H^+ the open half-plane of H bounded by the line through e_3 in which* **p** *lies,*
- **q** *γ-close to e_3 with $\gamma > 120°$,*
- *t_0 the unique equal-sided triangle incident to e_3 and located in H^+.*

Then the lines through the edges of t_0 different from e_3 decompose H^+ into four regions (Figure 9.1). Region 1 is t_0, region 2 that one which shares just one vertex with t_0, and regions 3 and 4 the two remaining ones. Regions 3 and 4 are symmetric and are treated in the same manner.
Let (P1), (P2) and (P3) be properties defined as

(P1) $l(\overline{\mathbf{pq}}) \leq \max\{l(\overline{\mathbf{pq}_1}), l(\overline{\mathbf{pq}_2})\}$.

(P2) $l(\overline{\mathbf{q}_1\mathbf{q}_2}) \leq \max\{l(\overline{\mathbf{pq}_1}), l(\overline{\mathbf{pq}_2})\}$.

(P3) $l(\overline{\mathbf{qq}_1}) \leq \max\{l(\overline{\mathbf{pq}_1}), l(\overline{\mathbf{pq}_2})\}$, $\quad l(\overline{\mathbf{qq}_2}) \leq \max\{l(\overline{\mathbf{pq}_1}), l(\overline{\mathbf{pq}_2})\}$.

Then

(A0) (P2) implies (P3).

(A1) In region 1, (P1) and not (P2).

(A2) In region 2, (P2) and not (P1).

(A3) In regions 3 and 4, (P1) holds. For the subregion of region 3 outside the disc around $\mathbf{q}_2$ *of radius $l(e_3)$, and for the subregion of region 4 outside the disc around* $\mathbf{q}_1$ *of radius $l(e_3)$, also (P2) holds.*

Proof: The radius of the circular arcs between $\mathbf{q}_1$ and $\mathbf{q}_2$ which bound the region of points $120°$-close to e_3 is equal to the length of e_3. The regions for $\gamma > 120°$ are subsets of this region.
If $\mathbf{q}$ is in the plane spanned by t, the assertions can be immediately concluded from Figure 9.1. A hint concerning (P1) is that the region of $120°$-close points is a subset of one of the discs bounded by the circle centered at $\mathbf{p}$ and running through $\mathbf{q}_1$ or through $\mathbf{q}_2$. This can be seen by considering the mutual locations of the bounding circular arc and the mentioned circles at $\mathbf{q}_1$ or $\mathbf{q}_2$.
If $\mathbf{q}$ is not in the plane spanned by t, the length of $\overline{\mathbf{pq}}$ is less than the length of $\overline{\mathbf{pq}'}$ where $\mathbf{q}'$ is obtained by rotating $\mathbf{q}$ around e_3 (Figure 9.2). ∎

An implication of the theorem is that the edge length of $\overline{\mathbf{pq}}$ may become larger than the lengths of the edges already in the graph G only in the case of region 2.
If $\mathbf{q}$ is not γ-close, but in the β_c-environment of e_3, the edge length can be estimated as follows.

Theorem 9.8 (Length bound for $\overline{\mathbf{pq}}$)
Let be

- *w a sector,*
- *H the plane spanned by $t(w)$,*
- *H^+ the open half-plane of H bounded by the line through e_3 in which* $\mathbf{p}$ *lies,*
- $\mathbf{q}$ *β_c-close to e_3 with $-1 \leq \beta_c \leq 1$.*

Then $l(\mathbf{p}, \mathbf{q}) \leq \min\{l(\overline{\mathbf{pq}_1}), l(\overline{\mathbf{pq}_2})\} + l(\overline{\mathbf{q}_1\mathbf{q}_2}) \leq \min\{l(\overline{\mathbf{pq}_1}), l(\overline{\mathbf{pq}_2})\} + l(\overline{\mathbf{pq}_1}) + l(\overline{\mathbf{pq}_2})$.

Proof: The first inequality follows from

$$l(\mathbf{p}, \mathbf{q}) = ||\mathbf{q} - \mathbf{p}|| \leq ||\mathbf{p} - \mathbf{q}_i|| + ||\mathbf{q}_i - \mathbf{q}|| \leq l(\overline{\mathbf{pq}_i}) + l(\overline{\mathbf{q}_1\mathbf{q}_2}),$$

with $i \in \{1, 2\}$. The relation $||\mathbf{q} - \mathbf{q}_i|| \leq l(\overline{\mathbf{q}_1\mathbf{q}_2})$ required in this inequality follows from $\mathbf{q} \in E_{\beta_c}(e_3)$. The second inequality of the theorem follows from the observation that the length of an edge of a triangle does not exceed the sum of the lengths of the other two edges of the triangle. ∎

The theorem tells that in the case of a β_c-close $\mathbf{q}$ which is not γ-close, the length of a possibly inserted edge $\overline{\mathbf{pq}}$ does increase by at most a factor of 3.

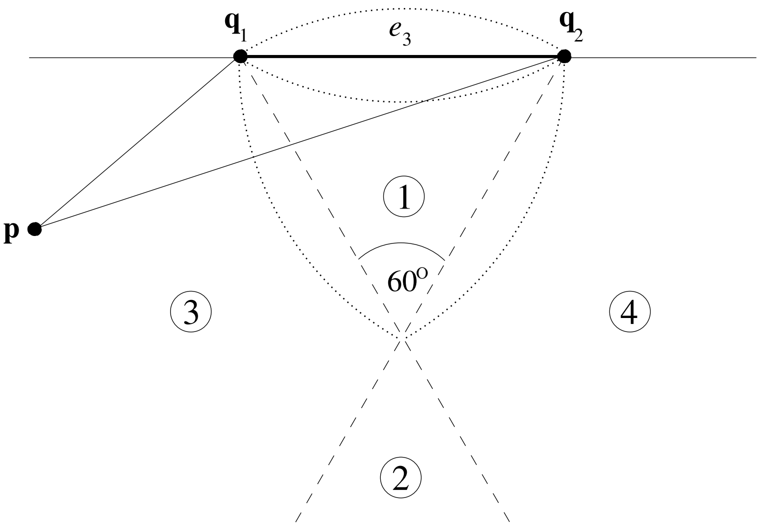


Figure 9.1: Analysis of the length of edges of type $\overline{\mathbf{pq}}$ with respect to a sector. The edges of an equal-sided triangle t_0 partition the half-plane H^+ into four regions for **q** with different properties of the length of $\overline{\mathbf{pq}}$.

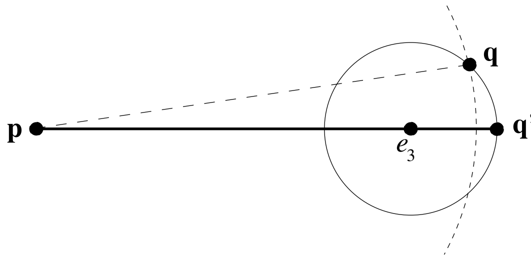


Figure 9.2: The length of $\overline{\mathbf{pq}}$ is less than the length between $\overline{\mathbf{pq}'}$ where $\mathbf{q}'$ is obtained by rotating **q** around e_3.

9.3 γ-Edges in the Flat Case

In the following we investigate the probability that a step of incremental edge insertion

- does not lead to the insertion of an edge at all, and
- does not yield a γ-edge,

in the case that the NN-images of all items involved in a decision fall into a flat part of the surface S, that is the NN-image of a line segment is the line segment itself, and all points involved are in a common plane. The importance of this case comes from the observation that for sufficiently short edges the environment of the surface which is involved in a decision is approximately flat.
For that purpose we analyze the different cases of edge insertion. The cases of the algorithm where edges may be inserted are the then-case B, the then-case D, and the else-case D.

9.3.1 Then-Case B

In the flat case, the candidate set $P_c^F(w)$ of a sector w consists of those sampling points which are in the interior of the triangle $t(w)$. The reason is that in this case the dihedral angles at the edges $\overline{\mathbf{qp}}$,

$\overline{\mathbf{qq}_1}$, and $\overline{\mathbf{qq}_2}$ are $180°$, and those at any other edge equal to $0°$. This property satisfies Definition 8.5.

Theorem 9.9 (Then-Case B) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case B fall into a flat part of S. If the candidate set $P_c^F(w)$ contains at least one point then a point $\mathbf{q} \in P_c^F(w)$ is found by the algorithm (Section 8.5.2) for which $e := \overline{\mathbf{pq}}$ does not NN-intersect any edge of G, and e is a γ-edge.*

Proof: Because all items involved in the test fall into a flat part of the surface, the points of $P_c^F(w)$ are identical with their projection on $t(w)$, that is, they are located in $t(w)$. Furthermore, $P_c^F(w)$ contains all sample points located in $t(w)$.

Let $\mathbf{q}$ be the point selected by the algorithm. Because G is NN-intersection-free, no edge of G NN-intersects e_1 and e_2. Because the region $t = \triangle(\mathbf{p}, \mathbf{q}_1', \mathbf{q}_2')$ of $t(w)$, $\mathbf{q}_1', \mathbf{q}_2'$ defined as in Section 8.5.2, is free of points, no edge is contained in this region. Thus no edge exists which might be NN-intersected by e, and thus e is NN-intersection-free.

Because e_1 and e_2 are γ-edges, no γ-close point for e can be found outside of $t(w)$. In order that a γ-close point can exist at all, the $\beta(\gamma)$-environment of e has to have a non-empty intersection with the region $t' := t(w) - t$. This implies that one of the angles between $\overline{\mathbf{pq}}$ and $\overline{\mathbf{q}_1'\mathbf{q}_2'}$ at $\mathbf{q}$ is less than $180° - \gamma$, and the other one is at least γ. Let w.l.o.g. $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{q}_1')$ be the triangle with angle γ at $\mathbf{q}$. This implies that in the triangle $\triangle(\mathbf{p}, \mathbf{q}, \mathbf{q}_1)$ the angle at $\mathbf{q}$ is at least γ, too. But this means that $\mathbf{q}$ is γ-close to e_1. This is a contradiction to the assumption that e_1 is a γ-edge. ■

9.3.2 Then-Case D

In the flat case, the then-case D becomes significantly simpler. The candidate set $P_{c,-}^{\beta_c}(w)$ of β_c-close points over a sector w becomes empty because all points in the interior of $t(w)$ belong to $P_c^F(w)$. This implies that the algorithm works as follows.

Step 1:
A candidate point $\mathbf{q} \in \overline{P}_{c,+}^{\beta_c}(w)$ is selected which encloses the largest angle with e_3.

Step 2, inside case:
This step is never entered because the candidate set is empty.

Step 2, outside case:
The final search is performed as in the original algorithm.

In the following we distinguish between the case that the candidate point $\mathbf{q}$ is γ-close to e_3, and the opposite case that $\mathbf{q}$ is not γ-close to e_3.

For the case that $\mathbf{q}$ is γ-close to e_3 we will first show that the probability that a fold-over occurs should be rather low. This observation implies that the point $\mathbf{q}$ selected during step 1 is the one with the largest enclosing angle with e_3, with high probability. Then we will show that under this condition the line segment $e := \overline{\mathbf{pq}}$ is intersection-free. This result implies that a replacement of $\mathbf{q}$ in search step 2, outside case, does not take place. The implication is that the point $\mathbf{q}$ selected in the condition of case D is a point with largest angle with e_3, with high probability.

The selected point $\mathbf{q}$ is used in the then-case D in order to decide which edge should be taken as candidate edge: e or e_3. We will show that the probability that e_3 is the candidate edge is low if e_3 is not a γ-edge. Furthermore we show that the probability that e is not a γ-edge is low, if $\mathbf{q}$ is γ-close to e_3.

An implication of the discussion is that for $\beta_c := \beta(\gamma)$ the probability is high that the resulting triangulation is NN-embeddable.

For the case that $\mathbf{q}$ is not γ-close to e_3 it turns out that e_3 is a γ-edge with high probability. The reason for possibly not using e_3 is the intention to reconstruct sharp edges or ridges. In that case β_c can be chosen between 0 and 1 so that $\beta(90°) = 0 \leq \beta_c \leq 1$.

If the surface is assumed free of such features, the choice $\beta_c := \beta(\gamma)$ will "switch off" this not required possibility. We do not investigate the case of sharp edges in detail because they do not occur for the SF-surfaces on which we have based our analysis.

9.3.2.1 Existence of fold-overs

In the plane, fold-over quadrilaterals can be characterized as follows.

Lemma 9.10 (Characterization of fold-over quadrilaterals) *Let w be a sector, and $\mathbf{q} \notin t(w)$ be a point in the plane spanned by w. Then the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ is fold-over in the sense of Definition 8.8 if and only if the points $\mathbf{p}$, $\mathbf{q}_1$, $\mathbf{q}$, $\mathbf{q}_2$ do not form a convex polygonal chain.*

Proof: The dihedral angles of interest for Definition 8.8 are either $180°$ or $0°$. The assertion of the theorem can easily be derived from this observation. ∎

Two types of fold-over quadrilaterals can be distinguished.

Definition 9.11 (Types of fold-over quadrilaterals) *A fold-over quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ in the plane has an* ***in-front fold-over*** *if $\mathbf{q}$ and $\mathbf{p}$ are on different sides of the line through $\mathbf{q}_1$ and $\mathbf{q}_2$. (Figure 9.3, left). Otherwise it is denoted as* ***back-fold-over***.

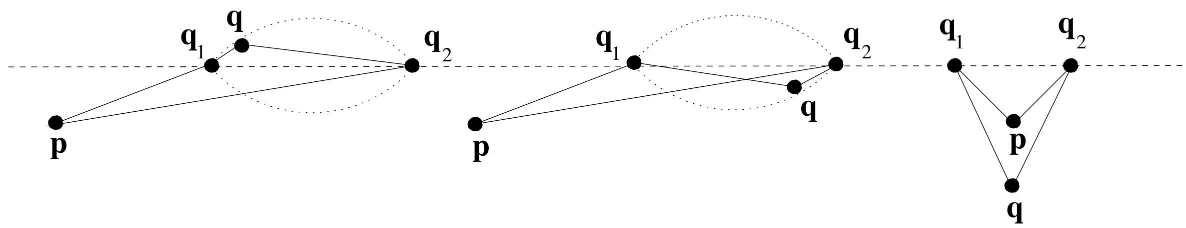


Figure 9.3: An in-front fold-over quadrilateral (left) and two examples of back-fold-over quadrilaterals. For the left and middle example additionally the boundary arcs of the $\beta(\gamma)$-environment are displayed (dotted lines).

The following theorem states that the configurations in which a back-fold-over may happen for $\mathbf{q}$ γ-close to e_3 do not occur under the constraints of our investigations if $\gamma_c' \leq \gamma$, where γ_c' is the angle bound of the algorithm controlling the generation of boundaries. This choice of γ and γ_c' is reasonable in order that the boundary detection can have a selective effect at all if $\beta_c = \beta(\gamma)$ is used. An implication of this choice of γ is that $\mathbf{p}$ is not γ-close to e_3, or, equivalently, that $\mathbf{p}$ is not in the $\beta(\gamma)$-environment of e_3.

Theorem 9.12 (Back-fold-over) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Let H be the plane spanned by $t(w)$, and H^+ be the open half-plane of H bounded by the line through e_3 in which*

p *lies. Let* $\mathbf{q} \in H^+ - t$ *be* γ*-close to* e_3*,* $90° \leq \gamma < 180°$*. Let* l_1 *and* l_2 *be the two lines through* $\mathbf{q}_1$ *and* $\mathbf{q}_2$*, respectively, so that the angle between* l_i*,* $i = 1, 2$*, and* e_3 *in* H^+ *is* $180° - \gamma$*. The lines induce three regions in* H^+ *: a region* A *defined by the wedge at* $\mathbf{q}_1$ *between* e_3 *and* l_1*, a region* B *defined by the wedge at* $\mathbf{q}_2$ *between* e_3 *and* l_2*, and a region* C *which is the rest of* H^+ *not covered by regions* A *and* B *(Figure 9.4). Then the following holds:*

(1) If **p** *is in region* A *or* B *and* **p** *is not in the* $\beta(\gamma)$*-environment of* e_3*, then* **q** *is in the* $\beta(\gamma)$*-environment of* e_1 *or* e_2*, and thus this case is impossible. If* $\gamma'_c \leq \gamma$ *where* γ'_c *is the angle bound of the algorithm controlling the generation of boundaries then* **p** *cannot be in the* $\beta(\gamma)$*-environment of* e_3*.*

(2) **p** *cannot be located in region* C*.*

Proof: Let **p** be in region A (region B analogously). If **p** is not in the $\beta(\gamma)$-environment of e_3, then the part of the $\beta(\gamma)$-environment in H^+ is a subset of the union of the triangle $t(w)$ and the $\beta(\gamma)$-environments of e_1 or e_2. Thus, because **q** is not in $t(w)$, **q** must be in the $\beta(\gamma)$-environment of e_1 or e_2. But e_1 and e_2 belong to G and are assumed to be γ-edges. Thus this case is not possible.

Because sectors with angles larger than γ'_c are not treated by the algorithm, **p** cannot be in the $\beta(\gamma)$-environment of e_3.

Let c be the circular arc defining the boundary of the $\beta(\gamma)$-environment in H^+. Then l_1 and l_2 are tangent to c. If **p** is in C, the part of the $\beta(\gamma)$-environment in H^+ is a subset of the triangle $t(w)$. Thus a **q** like in the theorem does not exist, and **p** cannot be located in C. ∎

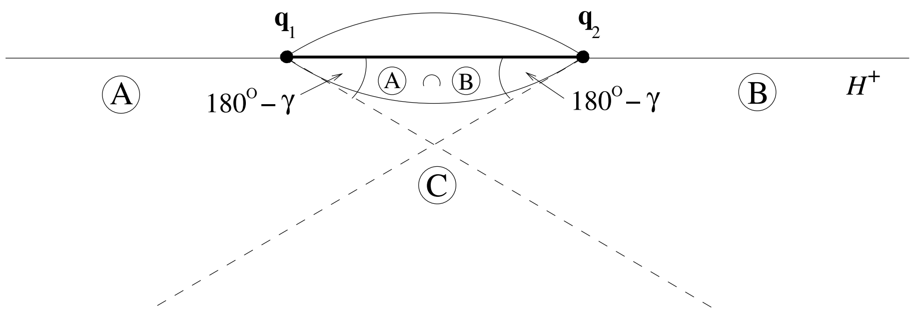


Figure 9.4: Illustration of the investigation of a back-fold-over of the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$. In this case **q** and **p** are located on the same side of the line of e_3.

The event of an in-front fold-over is treated in the following theorem.

Theorem 9.13 (In-front fold-over) *Let* S *be a surface,* P *be a sample set of* S*. Let the current graph* G *only consist of* γ*-edges which additionally are NN-intersection-free. Let* w *be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of* S*. Let* H *be the plane spanned by* $t(w)$*, and* H^+ *be the open half-plane of* H *bounded by the line through* e_3 *in which* **p** *lies.*

Let $\mathbf{q} \in H - H^+$ *be* γ*-close to* e_3*,* $90° \leq \gamma < 180°$*. Let* l_1 *and* l_2 *be the two lines through* $\mathbf{q}_1$ *and* $\mathbf{q}_2$*, respectively, so that the angle between* l_i*,* $i = 1, 2$*, and* e_3 *in* H^+ *is* γ*. The lines partition* H^+ *into three regions: a region* A *incident to* e_3*, and two symmetric regions* B *and* C *(Figure 9.5). Then the following holds:*

(1) If **p** *is in region A, then no fold-over occurs.*

(2) If **p** *is in region B or C, then a fold-over may occur.*

Proof: A fold-over does not occur if and only if the line of e intersects e_3. Let c be the circular arc defining the boundary of the $\beta(\gamma)$-environment in $H - H^+$. Then l_1 and l_2 are tangent to c at $\mathbf{q}_1$ and $\mathbf{q}_2$, respectively. Evidently, $\overline{\mathbf{pq}}$ intersects e_3 if **p** is in A.
If **p** is in B (analogously for C) then the line segment $\overline{\mathbf{pq}_1}$ is completely in B, and thus does not intersect e_3. Because $\overline{\mathbf{pq}_1}$ is not a tangent of c at $\mathbf{q}_1$ we can find a point **q** in the $\beta(\gamma)$-environment, for example in the neighborhood of $\mathbf{q}_1$, so that the line $\overline{\mathbf{pq}}$ does not intersect e_3, too. ■

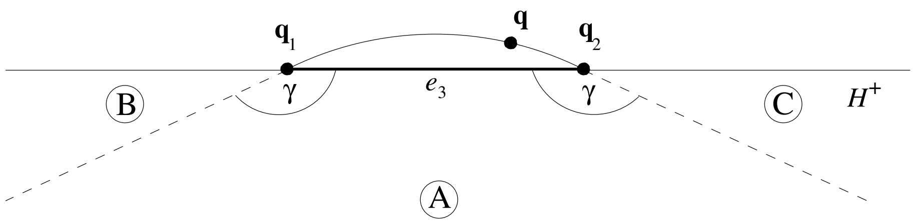


Figure 9.5: Configuration for which edge e_3 has to be the candidate edge of the then-case D, because of a fold-over of the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$, if **q** and **p** are located on different sides of the line of e_3.

The results of this section can be summarized as follows.

Observation 9.14 (Occurrence of fold-over quadrilaterals) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Let* $\mathbf{q} \in H - H^+$ *be γ-close to e_3, $90° \leq \gamma < 180°$. If the quadrilateral* $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ *is fold-over, then it is in-front foldover. The probability of occurrence of such a quadrilateral is low.*

Argumentation: Theorem 9.12 excludes back-fold-overs. Theorem 9.13 tells that in-front fold-over quadrilaterals may occur. However, their occurrence is restricted to a small region of possible locations of **p** which decreases towards 0 with increasing γ. ■

9.3.2.2 Intersection-freeness

For the case that **q** is γ-close to e_3 the following theorem shows that the line segments e_3 and $e := \overline{\mathbf{pq}}$ both are intersection-free under the assumption of γ-edges and local flatness.

Theorem 9.15 (NN-intersection-freeness in the then-case D) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Let* **q** *be a point γ-close to e_3 and with largest angle with e_3. Then the edge e_3 and the edge $e =$* $\overline{\mathbf{pq}}$ *do not NN-intersect any edge of G.*

Proof:

Case $e' = e_3$: For flat surface regions, every edge which NN-intersects e_3 enters $t(w)$, and either ends at a vertex in $t(w)$ or leaves $t(w)$ at edge e_1 or e_2. But all that is impossible for the following reasons. None of the vertices is in $t(w)$ because case D is in the else-case B. Furthermore the edges e_1 and e_2 do not NN-intersect any edge by assumption. Thus e_3 is NN-intersection-free.

Case $e' = e$**:** By the same arguments as in the proof of Theorem 9.9, the NN-intersection of triangle $t(w)$ with any edge of G is empty. Thus $\overline{\mathbf{pq}}$ might only be intersected in the $\beta(\gamma)$-environment outside of t. Because $\mathbf{q}$ has been assumed to be the γ-close point to e_3 with the largest angle (Section 8.7), an NN-intersecting edge e'' of G intersects the boundary of that region twice, and separates $\mathbf{q}$ from e_3 (Figure 9.6). But that implies that $\mathbf{q}$ is γ-close to e'', in contradiction to the assumption. ∎

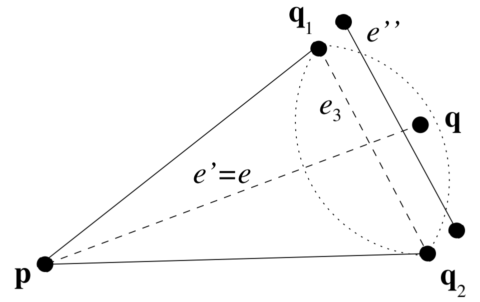


Figure 9.6: An edge e'' possibly intersecting e', intersects the circular boundary of the $\beta(\gamma)$-environment of e_3 twice. From this configuration it can be concluded that $\mathbf{q}$ is also in the $\beta(\gamma)$-environment of e'' because it is γ-close to e_3.

9.3.2.3 Emptiness of the search of step 2, outside case

The preceding theorem is now used to show that step 2, outside case, is not executed under the restrictions imposed in the current discussion. Additionally, the following lemma is required.

Lemma 9.16 *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Let* $\mathbf{q}$ *be a sample point with largest angle to e_3, which does not induce a fold-over. Let $e'' := \overline{\mathbf{q}_1\mathbf{r}}$ (analogously $\overline{\mathbf{q}_2\mathbf{r}}$) be an edge of G so that* $\mathbf{p}$ *and* $\mathbf{r}$ *are separated by the line through e_3. Then e'' intersects $e = \overline{\mathbf{pq}}$ or the angle between e'' and e_3 at $\mathbf{q}_1$ is larger than the angle between $e' := \overline{\mathbf{q}_1\mathbf{q}}$ and e_3.*

Proof: Let ξ' be the angle between e' and e_3, and ξ'' be the angle between e'' and e_3.
Let c be the arc through $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}$. If the ray with origin $\mathbf{q}_1$ through $\mathbf{r}$ does not intersect c then ξ'' is clearly larger than ξ'. If the ray intersects c on the segment between $\mathbf{q}_1$ and $\mathbf{q}$, then it is again immediately clear that $\xi'' > \xi'$. If the ray intersects c on the segment between $\mathbf{q}$ and $\mathbf{q}_2$, then the ray intersects $\overline{\mathbf{pq}}$ because $\overline{\mathbf{pq}}$ intersects e_3 since fold-over-freeness is assumed. Because $\mathbf{q}$ is the sample point with largest angle, not just the ray, but also the line segment e'' intersects e', because $\mathbf{r}$ is outside the region bounded by e_3 and c. Figure 9.7 illustrates the discussion. ∎

Now we obtain another theorem.

Theorem 9.17 (Emptiness of the search of step 2, outside case) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Let* $\mathbf{q}$ *be a sample point γ-close to e_3 and with largest angle to e_3, which does not induce a fold-over. Then the search of step 2, outside case, is empty.*

Proof: By Lemma 9.16, $\overline{\mathbf{pq}}$ intersects or the angles of the edges under consideration are so that $\mathbf{q}$ is not updated. Because intersection is excluded by Theorem 9.15 in the assumed case of $\mathbf{q}$ γ-close to e_3, the second alternative holds, that an update does not take place. ∎

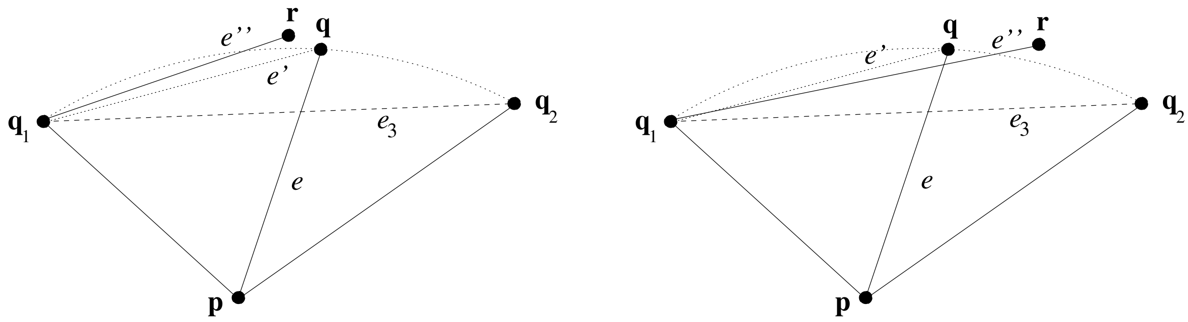


Figure 9.7: The two possible configurations of a line segment $\overline{\mathbf{pr}}$, $\mathbf{r}$ a candidate point for replacing $\mathbf{q}$.

By using the result on the unlikeliness of fold-overs, we can derive the following observation.

Observation 9.18 (Emptiness of the search of step 2, outside case) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Let* $\mathbf{q}$ *be a sample point γ-close to e_3 and with largest angle to e_3. Then with high probability the search of step 2, outside case, is empty.*

Argumentation: Because with high probability $\mathbf{q}$ does not induce a fold-over quadrilateral the assertion of Theorem 9.17 which is just that of the observation, holds with high probability. ■

9.3.2.4 γ-Edge Property

Reasons for the event that a non-γ-edge might be inserted are

- e_3 is the candidate edge resulting from the min-max triangulation,
- $\overline{\mathbf{pq}}$ is the candidate edge resulting from the min-max triangulation, and it is not a γ-edge.

In the following we describe the configurations in which these events may happen. It turns out that they are very special and thus the probability of their occurrence is low.

Edge e_3 is selected as candidate edge by the min-max triangulation if a fold-over occurs, or if no fold-over occurs but the angle of the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ at point $\mathbf{q}_1$ or at point $\mathbf{q}_2$ is larger than the angle at $\mathbf{q}$.

As we know from Observation 9.14 the probability of over-folding quadrilaterals is low so that we can exclude this case from the discussion.

In order that the min-max triangulation yields e_3, the angle at point $\mathbf{q}_1$ or at point $\mathbf{q}_2$ of a non-over-folding quadrilateral has to be larger than the angle at $\mathbf{q}$. In this case we get Theorem 9.19.

Theorem 9.19 (Non-over-folding quadrilateral with large angle at $\mathbf{q}_1/\mathbf{q}_2$) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Let H be the plane spanned by $t(w)$, and H^+ be the open half-plane of H bounded by the line through e_3 in which* $\mathbf{p}$ *lies. Let $\mathbf{q} \in H - H^+$ be γ-close to e_3, $90° \leq \gamma < 180°$. Let l_1 and l_2 be the two lines through $\mathbf{q}_1$ and $\mathbf{q}_2$, respectively, so that the angle between l_i, $i = 1, 2$, and e_3 in H^+ is $2\gamma - 180°$. The lines partition H^+ into three regions: a region A incident to e, and two symmetric regions B and C (Figure 9.8). Then the following holds:*

(1) If **p** *is in region* A *and* $\mathbf{q} \in H - H^+$*, then the angle at* $\mathbf{q}_1$ *or* $\mathbf{q}_2$ *in the quadrilateral* $\Box(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ *is less than* γ*, and thus less than the angle at* **q**.

(2) If **p** *is in region* B *or* C*, and* $\mathbf{q} \in H - H^+$*, then the angle at* **q** *in the quadrilateral* $\Box(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ *might be less than the angle at* $\mathbf{q}_1$ *or* $\mathbf{q}_2$.

Proof: If **p** is in region A and $\mathbf{q} \in H - H^+$, then the angle at $\mathbf{q}_1$ is less than the sum of the angle at $\mathbf{q}_1$ ($\mathbf{q}_2$) between l_1 (l_2) and e_3, and the angle between e_3 and $\overline{\mathbf{q}_1\mathbf{q}}$, that is less than $(2\gamma - 180°) + (180° - \gamma) = \gamma$. Because the angle at **q** is at least γ, assertion (1) holds.
Because the line $\overline{\mathbf{pq}_1}$ (or $\overline{\mathbf{pq}_2}$) traverses the part of the $\beta(\gamma)$-environment in $H - H^+$, the angle at $\mathbf{q}_1$ or $\mathbf{q}_2$ can be up to $180°$, and thus is larger than the angle at **q** which always is less than $180°$. ■

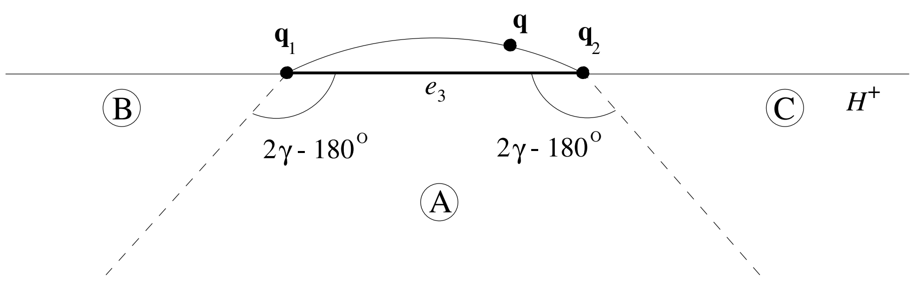


Figure 9.8: Configuration for which edge e_3 has to be the candidate edge of the then-case D, because of a non-over-folding quadrilateral $\Box(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ with larger angle at $\mathbf{q}_1$ or $\mathbf{q}_2$ than at **q**.

The implication of the theorem is that the probability that the min-max triangulation has to decide for e_3 is low because its corresponding regions are B and C. If γ increases towards $180°$ then the area of B and C relative to the whole area decreases to 0.
The second event is that $\overline{\mathbf{pq}}$ is reported as result of the min-max triangulation. The configurations of this event concerning the property of being a γ-edge can be characterized as follows.

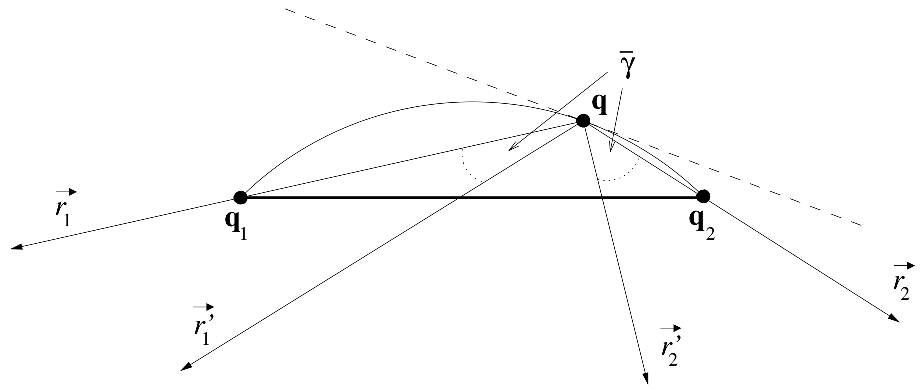


Figure 9.9: If $e := \overline{\mathbf{pq}}$ is not a γ-edge then **p** is located in the wedge between $\vec{r_1}$ and $\vec{r'_1}$, or in the wedge between $\vec{r_2}$ and $\vec{r'_2}$.

Theorem 9.20 (Non-γ-edges $\overline{\mathbf{pq}}$) *Let* S *be a surface,* P *be a sample set of* S*. Let the current graph* G *only consist of* γ*-edges which additionally are NN-intersection-free. Let* w *be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part* S*. Let*

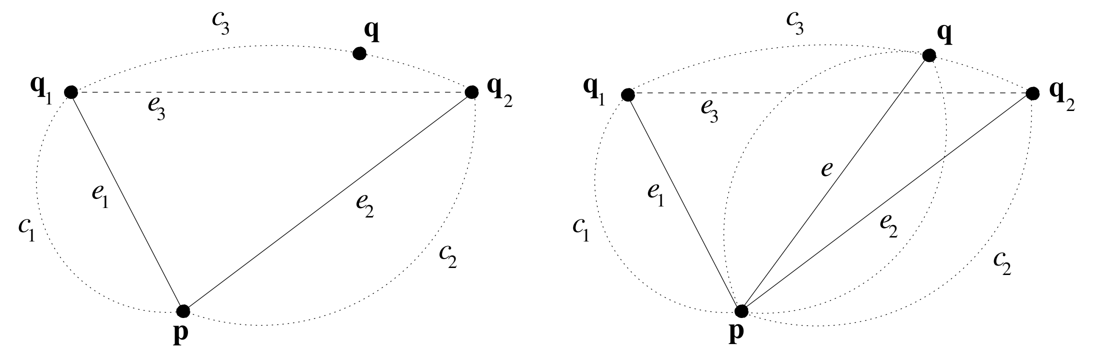


Figure 9.10: The region $c(w)$ (left) and the location of $E_{\beta(\gamma)}(e)$ in $c(w)$.

(1) $\mathbf{q} \in H - H^+$ *(H, H^+ like in Theorem 9.19) be γ-close to e_3, with largest angle,*

(2) the quadrilateral $\square(\mathbf{p}, \mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$ *be non-fold-over,*

(3) $\overrightarrow{r_1}$ *and* $\overrightarrow{r_2}$ *be the two rays at* $\mathbf{q}$ *through* $\mathbf{q}_1$ *and* $\mathbf{q}_2$*, respectively,*

(4) $\overrightarrow{r'_1}$ *and* $\overrightarrow{r'_2}$ *be the two rays at* $\mathbf{q}$ *intersecting* e_3 *and with angle* $\overline{\gamma} := 180° - \gamma$ *to the lines* $\overline{\mathbf{qq}_1}$ *and line* $\overline{\mathbf{qq}_2}$*, respectively (cf. Figure 9.9).*

If $e := \overline{\mathbf{pq}}$ *is not a γ-edge then* $\mathbf{p}$ *is located in the wedge between* $\overrightarrow{r_1}$ *and* $\overrightarrow{r'_1}$*, or in the wedge between* $\overrightarrow{r_2}$ *and* $\overrightarrow{r'_2}$*.*

Proof: Let γ' be the angle at $\mathbf{q}$ of the triangle $\triangle(\mathbf{q}_1, \mathbf{q}, \mathbf{q}_2)$. The union $c(w)$ of $t(w)$, $E_{\beta(\gamma)}(e_1)$, $E_{\beta(\gamma)}(e_2)$, and $E_{\beta(\gamma')}(e_3)$ is bounded by three circular arcs c_1, c_2, and c_3 corresponding to the edges e_1, e_2, and e_3 (Figure 9.10). In the else-case D, $t(w)$ is free of sample points. $E_{\beta(\gamma)}(e_1)$ and $E_{\beta(\gamma)}(e_2)$ are free of sample points because the edges of G are γ-edges. $E_{\beta(\gamma')}(e_3)$ is free of sample points because $\mathbf{q}$ has a largest angle by (1). Thus $c(w)$ is free of sample points.

Because $c(w)$ is free of sample points, $E_{\beta(\gamma)}(e)$ has to intersect the boundary of $c(w)$, that is c_1, c_2, or c_3.

If $E_{\beta(\gamma)}(e)$ intersects c_1 (c_2 can be treated analogously), we claim that $\mathbf{q}_1$ is in $E_{\beta(\gamma)}(e)$. Furthermore the boundary arcs of $E_{\beta(\gamma)}(e)$ are separated from $\mathbf{q}_1$ by the rays $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ (Figure 9.9) because the tangents of the two boundary arcs at $\mathbf{q}$ have an angle of $\overline{\gamma}$ with e, and the quadrilateral does not fold-over. Thus $\mathbf{q}_1$ cannot be in $E_{\beta(\gamma)}(e)$ if $\mathbf{p}$ is in the wedge between $\overrightarrow{r'_1}$ and $\overrightarrow{r'_2}$, or vice versa, $\mathbf{p}$ has to be in one of the wedges of the theorem.

In order to prove the claim, we rotate e_1 together with $E_{\beta(\gamma)}(e_1)$ onto e. The arc c'_1 of the resulting $E_{\beta(\gamma)}(\overline{\mathbf{pq}'_1})$ does not intersect c_1. If we move $\mathbf{q}'_1$ towards $\mathbf{p}$, then c'_1 does not intersect, too. Thus $\mathbf{q}'_1$ has to be moved into the opposite direction in order to reach $\mathbf{q}$. But the first intersection of c'_1 with c_1 which may happen is at $\mathbf{q}_1$. The second arc c'_2 of $E_{\beta(\gamma)}(\overline{\mathbf{pq}'_1})$ cannot intersect c_1 at all. Both together proves the claim.

If $\mathbf{p}$ is in the wedge between $\overrightarrow{r'_1}$ and $\overrightarrow{r'_2}$, the case that $E_{\beta(\gamma)}(e)$ intersects c_3 does not occur because the bounding arcs of $E_{\beta(\gamma)}(e)$ are separated from c_3 by the rays $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ (Figure 9.9). The reason of separation is that the tangents of the two boundary arcs at $\mathbf{q}$ have an angle of $\overline{\gamma}$ with e, and the quadrilateral does not fold-over. Thus in order that $E_{\beta(\gamma)}(e)$ intersects c_3, $\mathbf{p}$ has to be in one of the wedges of the theorem. ∎

In summary we obtain a new observation.

Observation 9.21 (Occurrence of non-γ-edges) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the then-case D fall into a flat part of S. Then the following holds:*

(1) The probability that a non-γ-edge e_3 is reported by the then-case D is low.

(2) If an edge $e := \overline{\mathbf{pq}}$ with $\mathbf{q}$ γ-close to e_3 is selected by the then-case D then the probability that e is not a γ-edge is low.

Argumentation: e_3 is selected if

- a fold-over occurs, or
- if no fold-over occurs and the selected point **q** is γ-close to e_3.

The probability of the first case is low according to Observation 9.14, and the probability of the second case is low by Theorem 9.19. Thus (1) holds.
(2) is an immediate implication of Theorem 9.20. ■

9.3.3 Else-Case D

The else-case D becomes active if

- the candidate set $P_c^{\beta_c}(w)$ is empty, or
- the candidate set $P_c^{\beta_c}(w)$ consists just of points **q** which induce a fold-over quadrilateral, or for which $e = \overline{\mathbf{pq}}$ intersects an already existing edge.

In the first case we get the following lemma.

Lemma 9.22 *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the else-case D fall into a flat part of S. Let the candidate set $P_c^{\beta_c}(w)$ be empty. Then the edge e_3 is a γ-edge and does not NN-intersect any edge of G.*

Proof: e_3 is a γ-edge because the candidate set $P_c^{\beta_c}(w)$ is empty. For intersection-freeness of e_3 it can be argued as in the proof of Theorem 9.15. ■

In the second case we get Observation 9.23.

Observation 9.23 *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the else-case D fall into a flat part of S. Let the candidate set $P_c^{\beta_c}(w)$ consist just of points* **q** *which induce a fold-over quadrilateral, or for which $e = \overline{\mathbf{pq}}$ intersects an already existing edge. Then with high probability e_3 is a γ-edge and does not NN-intersect any edge of G.*

Argumentation: We know from Observation 9.14 that the probability of an over-folding quadrilateral can be considered as low. Furthermore, if e intersects an already existing edge then **q** cannot be γ-close to e_3 with largest angle (Theorem 9.15). In order that such a **q** would not have been selected in the then-case D,

- the point with largest angle would not have been γ-close,
- or the point with largest angle would have been γ-close but causing a fold-over.

The probability of the second case is low by Observation 9.14. Thus the only case which remains is the first one which means that e_3 does not have a γ-close point with high probability.
For intersection-freeness of e_3 it can be argued as in the proof of Theorem 9.15. ∎

The following observation summarizes these results.

Observation 9.24 (γ-edge property of else-case D) *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the test of the else-case D fall into a flat part of S. Then with high probability, e_3 is a γ-edge and does not NN-intersect any edge of G.*

Argumentation: Because the two conditions at the beginning of the section cover the possibilities that the then-case D is entered, Lemma 9.22 and Observation 9.23 imply this observation. ∎

9.3.4 Summary

The investigations of this section can be summarized as follows.

Observation 9.25 *Let S be a surface, P be a sample set of S. Let the current graph G only consist of γ-edges, $90° \leq \gamma < 180°$, which additionally are NN-intersection-free. Let w be a sector for which the NN-images of all items involved into the tests of the then-case B, the then-case D, and the else-case D fall into a flat part of S. Let $\beta_c := \beta(\gamma)$.*

(1) The only reason that no edge might be inserted by the algorithm is that the candidate edge intersects an already existing edge of G. This event does not occur.

(2) For the then-case B the candidate edge is always a γ-edge. For the other cases the probability that the candidate edge is a γ-edge is high.

Argumentation:
(1) summarizes Theorems 9.9, 9.15, and 9.24.
(2) is a conclusion from Theorem 9.9 and the Observations 9.21 and 9.24. ∎

9.4 γ-Edges in the Curved Case

The idea of the analysis of the curved case is to compare the treatment of a sector w on the given set P of sampling points and the current graph G with the treatment of the sector in the orthogonal projection of P and G in a local environment of w onto the tangent plane of the surface at the sector center $\mathbf{p}$. The environment comprehends the part of G which is relevant for processing w. We will argue that the projection is a one-to-one mapping between the current manifold M in space and its projection, that G and its projection have the same behavior with respect to γ-edges, and that the algorithm behaves in the same manner in both cases, all this with high probability.

The background of this approach is that for a sufficiently small environment of a point $\mathbf{p}$ on an SF-surface S, the deviation between the tangent plane of S at $\mathbf{p}$ and the surface S is very small. This is quantified in the following Lemma which shows that perturbations of vertices caused by projection on a suitable plane are small.

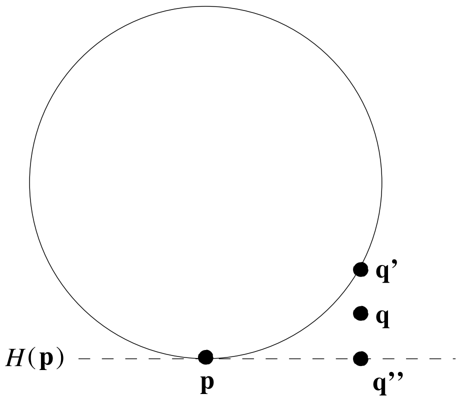


Figure 9.11: Illustration of the proof of Lemma 9.26.

Lemma 9.26 (Approximation by tangential planes) *Let S be a compact SF-surface without boundary with an SF-radius r, $H(\mathbf{p})$ the tangential plane at a point $\mathbf{p} \in S$. Let $l(\mathbf{p},\mathbf{q})$ be the Euclidean distance of a point $\mathbf{q} \in S$ from $\mathbf{p}$, $h(\mathbf{p},\mathbf{q})$ the Euclidean distance of $\mathbf{q}$ from $H(\mathbf{p})$. Then*

$$\lim_{l(\mathbf{p},\mathbf{q})\to 0} \frac{h(\mathbf{p},\mathbf{q})}{l(\mathbf{p},\mathbf{q})^2} \leq \frac{1}{r}.$$

Equivalently, for every $\varepsilon > 0$ a $\delta > 0$ exists so that for $l(\mathbf{p},\mathbf{q}) < \delta$, $\mathbf{p} \neq \mathbf{q}$,

$$\frac{h(\mathbf{p},\mathbf{q})}{l(\mathbf{p},\mathbf{q})^2} \leq \frac{1}{r} + \varepsilon.$$

Additionally, δ can be chosen globally independent from $\mathbf{p}$.

Proof: Because S is an SF-surface, a pair of tangential spheres exists at $\mathbf{p}$. S is located between those spheres and does not intersect them. Furthermore, the spheres are tangential to $H(\mathbf{p})$. Thus the distance of a point $\mathbf{q} \in S$ close to $\mathbf{p}$ from $H(\mathbf{p})$ is at most the distance of a point $\mathbf{q}'$ on one of the spheres which orthogonally projects onto the same point $\mathbf{q}''$ on $H(\mathbf{p})$ as $\mathbf{q}$,

$$h(\mathbf{p},\mathbf{q}) = l(\mathbf{q},\mathbf{q}'') \leq l(\mathbf{q}',\mathbf{q}''),$$

cf. Figure 9.11. The distance of a point $\mathbf{q}'$ on the sphere from $H(\mathbf{p})$ is

$$l(\mathbf{q}',\mathbf{q}'') = r - \sqrt{r^2 - l(\mathbf{p},\mathbf{q}'')^2},$$

where r is the radius of the sphere. The distance of $\mathbf{q}''$ from $\mathbf{p}$ is at most the distance of $\mathbf{q}$ from $\mathbf{p}$,

$$l(\mathbf{p},\mathbf{q}'') \leq l(\mathbf{p},\mathbf{q}).$$

From these relations we get

$$\lim_{l(\mathbf{p},\mathbf{q})\to 0} \frac{h(\mathbf{p},\mathbf{q})}{l(\mathbf{p},\mathbf{q})^2} \leq \lim_{l(\mathbf{p},\mathbf{q}'')\to 0} \frac{l(\mathbf{q}',\mathbf{q}'')}{l(\mathbf{p},\mathbf{q}'')^2} = \lim_{l(\mathbf{p},\mathbf{q}'')\to 0} \frac{2l(\mathbf{p},\mathbf{q}'')}{2l(\mathbf{p},\mathbf{q}'')\sqrt{r^2 - l(\mathbf{p},\mathbf{q}'')^2}} = \frac{1}{r}.$$

The re-formulation with ε and δ is just the definition of the limit.
δ is global because if δ is chosen so that

$$\frac{l(\mathbf{q}',\mathbf{q}'')}{l(\mathbf{p},\mathbf{q}'')^2} \leq \frac{1}{r} + \varepsilon$$

for $l(\mathbf{p}, \mathbf{q}'') < \delta$, then δ is independent from the location of $\mathbf{p}$ on the surface: it just depends on the surface-independent configuration depicted in Figure 9.11. Because $l(\mathbf{p}, \mathbf{q}'') \leq l(\mathbf{p}, \mathbf{q})$ we get

$$\frac{h(\mathbf{p}, \mathbf{q})}{l(\mathbf{p}, \mathbf{q})^2} \leq \frac{l(\mathbf{q}', \mathbf{q}'')}{l(\mathbf{p}, \mathbf{q}'')^2} \leq \frac{1}{r} + \varepsilon$$

for $l(\mathbf{p}, \mathbf{q}) < \delta$. ■

A crucial concept for the following is the tangent plane environment of points which is defined as follows (Figure 9.12).

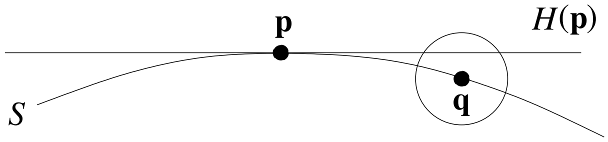


Figure 9.12: Depiction of the tangent-plane environment (TPE) of $\mathbf{q}$.

Definition 9.27 (Tangent plane environment (TPE)) *Let S be a compact SF-surface without boundary with SF-radius r, $l(\mathbf{p}, \mathbf{q})$ be the distance of a point $\mathbf{q} \in S$ from $\mathbf{p}$, and $\varepsilon > 0$.*
*Then the (ε-)**tangent plane environment (TPE)** of $\mathbf{q}$ with respect to $\mathbf{p}$ is defined as the closed sphere of radius $(\frac{1}{r} + \varepsilon)l(\mathbf{p}, \mathbf{q})^2$.*

Corollary 9.28 (Tangent plane intersection property of TPEs) *Let S be a compact SF-surface without boundary with SF-radius r, $H(\mathbf{p})$ be the tangential plane at a point $\mathbf{p} \in S$, $l(\mathbf{p}, \mathbf{q})$ be the distance of a point $\mathbf{q} \in S$ from $\mathbf{p}$, and $\varepsilon > 0$. Then a $\delta > 0$ exists so that for $l(\mathbf{p}, \mathbf{q}) < \delta$ the ε-TPE of $\mathbf{q}$ intersects $H(\mathbf{p})$.*

Proof: The corollary is a re-formulation of Lemma 9.26 using the preceding definition. ■

In the following, statements are made, using this corollary, which hold under the condition of "a sufficiently small environment $E(\mathbf{p})$ of $\mathbf{p}$, and for sufficiently small edge lengths of the manifold". The reason of the condition is to guarantee that the vertices of the part of the graph G involved in the treatment of a sector by the algorithm satisfy the corollary. The details are as follows. The "sufficiently small environment" is the one of all $\mathbf{q} \in S$ with $l(\mathbf{p}, \mathbf{q}) < \delta$ in the sense of the corollary. The "sufficiently small edge lengths" are edge lengths which are less than $c \cdot \delta$ where $0 < c < 1$ is a constant factor. The factor has the property that all vertices or edges involved in the treatment of a sector w centered at $\mathbf{p}$ by the algorithm are contained in $E(\mathbf{p})$. Because the distances of all single vertices and all vertices of edges involved from the sector triangle $t(w)$ can be expected to be less than a constant factor of the edge length bound. The edge length bound multiplied by a constant factor has to be bounded by δ in order to have all involved edges and vertices in $E(\mathbf{p})$. But this means that the edge lengths are bounded by a constant fraction c of δ.

The observations made in the following usually hold "with high probability". In the argumentation the problematic cases are reduced to special configurations of points. One special configuration consists of three points which are approximately co-linear. Another example are three points on a circle which has an additional constraint on the radius. In the space of all configurations of triples of points these configurations have the measure zero if all configurations are equally weighted. This can be expected if the sampling points are uniformly distributed. It also can be achieved if the sampling strategy is chosen so that sampling points close to each other do not have these special patterns, similar to conditions "no three input points are co-linear" or "no four points a co-circular" often imposed on the input of geometric algorithms.

Observation 9.29 *Let S be a compact SF-surface without boundary, $H(\mathbf{p})$ be the tangent plane of S at $\mathbf{p} \in S$, P be a finite sampling set, M be an intermediate manifold constructed by the algorithm without χ_c-intersecting edges and points flat over a triangle. Then for a sufficiently small environment $E(\mathbf{p})$ of $\mathbf{p}$, and for sufficiently small edge lengths of the manifold, the projection $M'|_{E(\mathbf{p})}$ of the restriction $M|_{E(\mathbf{p})}$ of M to $E(\mathbf{p})$ does not have self-overlappings, with high probability.*

Argumentation: We assume that $M'|_{E(\mathbf{p})}$ has a self-overlapping, that is two of its objects (edges or faces) intersect. Two such objects of the planar manifold $M'|_{E(\mathbf{p})}$ intersect if two edges intersect or one of the objects is completely contained in the other object.

If the edge lengths are small, then the TPEs of the involved vertices are small compared to the length of the involved edges. Furthermore, by definition, all TPEs intersect the tangential plane. Thus, if two projected edges intersect, then either the original edges can be assumed to χ_c-intersect, or three of the four involved original vertices are approximately co-linear. Since the triangulation algorithm avoids χ_c-intersecting edges, and thus such edges are excluded in the formulation of the observation, and the probability of three co-linear vertices can be considered as low, two projected edges cannot intersect with high probability.

If an edge is a subset of a second edge, the four original vertices are approximately co-linear, because their TPEs all intersect the projection plane. Because the probability of this event can be assumed as low, this case does not occur with high probability.

If a projected triangle is a subset of an other triangle, one of its original vertices is flat over the second original triangle, or one of its original vertices is co-linear with the vertices of an original edge of the second triangle. The reason is that TPEs intersect the projection plane. Because the first case should be avoided by the algorithm and thus is excluded in the formulation of the observation, and because the probability of the second case can be assumed as low, this case does not happen, too.

Thus the assumption of the observation holds. ∎

Observation 9.30 (Preservation of γ-edge property under projection) *Let S be a compact SF-surface without boundary, $H(\mathbf{p})$ be the tangent plane of S at $\mathbf{p} \in S$, P be a finite sampling set, G be an intermediate graph constructed by the algorithm. Then for a sufficiently small environment $E(\mathbf{p})$ of $\mathbf{p}$, and for sufficiently small edge lengths of G, the projection $G'|_{E(\mathbf{p})}$ of the restriction $G|_{E(\mathbf{p})}$ of G, the projection of a γ-edge of $G|_{E(\mathbf{p})}$ is a γ-edge of the projection $G'|_{E(\mathbf{p})}$, with high probability.*

Argumentation: We assume that an edge $e = \overline{\mathbf{rs}}$ exists which is a γ-edge of $G|_{E(\mathbf{p})}$ but a non-γ-edge of $G'|_{E(\mathbf{p})}$ in its projection e'. In the following we show that in this case a sampling point $\mathbf{t}$ exists so that $\mathbf{r}$, $\mathbf{s}$, and $\mathbf{t}$ are approximately on a common circle of a radius dependent on γ. Because the probability of this event can be considered low, the probability that the assumption holds is low, and thus the probability of the assertion of the observation is high.

Because e' is not a γ-edge, a sampling point $\mathbf{t}$ exists whose projection $\mathbf{t}'$ is in the $\beta(\gamma)$-environment of $e' = \overline{\mathbf{r}'\mathbf{s}'}$. On the other hand, because e is a γ-edge, $\mathbf{t}$ is outside the $\beta(\gamma)$-environment of e. If the radii of the TPEs are small compared to the length of e, this implies that $\mathbf{t}$ has a distance from the boundary of the $\beta(\gamma)$-environment which is bounded by a small constant of the largest of these radii (Figure 9.13). But this means that $\mathbf{t}$ is approximately on a circle in the plane spanned by $\mathbf{r}$, $\mathbf{s}$, and $\mathbf{t}$, through $\mathbf{r}$ and $\mathbf{s}$, and the radius of the circles of the definition of the $\beta(\gamma)$-environment. Thus these three points are approximately on a common circle with a specific radius. ∎

The following observation states the equivalence of the intersection status of line segments in the spatial and the projected case, so that results on this aspect in the locally flat case can be immediately transferred to the spatial case, and thus can be excluded from further analysis of the algorithm.

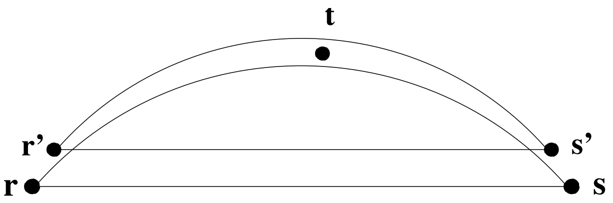


Figure 9.13: Illustration of the argumentation of Observation 9.30. **r**,**s**, and **t** are approximately on a common circle of a radius determined by γ.

Observation 9.31 (Preservation of intersection properties of edges) *Let S be a compact SF-surface without boundary, $H(\mathbf{p})$ be the tangent plane of S at $\mathbf{p} \in S$, P be a finite sampling set. Then for a sufficiently small environment $E(\mathbf{p})$ of $\mathbf{p}$, and for sufficiently small edge lengths, two line segments k and l in $E(\mathbf{p})$ with vertices in P χ_c-intersect if and only if their projections k' and l' intersect, with high probability.*

Argumentation: Because k and l are subsets of $E(\mathbf{p})$, the TPE-environments of the vertices of k and l intersect $H(\mathbf{p})$. k and l are obtained by slight perturbation of the vertices of k' and l', within a distance bounded by the diameters of the TPEs of these vertices. If k' and l' intersect, a danger that k and l do not intersect only arises if a vertex of k' or l' is close to l' or k', respectively (Figure 9.14). Closeness is determined by the TPE-diameters and the definition of χ_c-intersection. But this means that the mentioned vertex and the vertices of the mentioned line are approximately co-linear. But this event should have low probability.

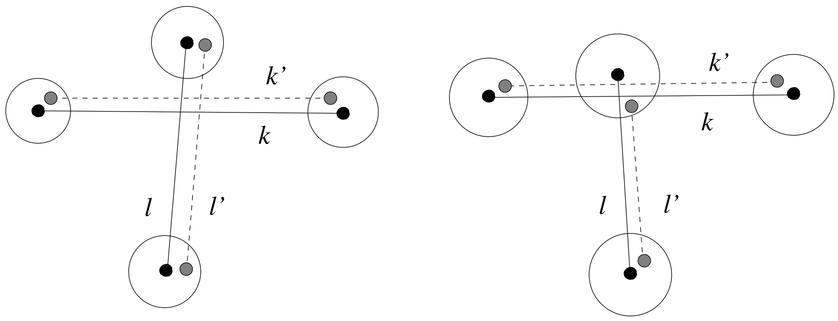


Figure 9.14: Illustration of the argumentation of Observation 9.31. The left configuration has an intersection in the original and in the projected case. The right configuration has an intersection in the original case, but not in the projected case. In the second situation these points are approximately co-linear.

If k' and l' do not intersect, k and l may be recognized as χ_c-intersecting only if a vertex of k' or l' is close to l' or k', respectively. As before, this means that the mentioned vertex and the vertices of the mentioned line are approximately co-linear. But this event should have low probability. ■

Observation 9.32 (γ-edge behavior of the algorithm) *Let S be a compact SF-surface without boundary, P be a finite sampling set. Let the current graph G only consist of γ-edges, $90^\circ \leq \gamma < 180^\circ$, which additionally are χ_c-intersection-free. Let all points of P which are flat over $t(w)$ be in the NN-image of $t(w)$. Let $\beta_c := \beta(\gamma)$. If the lengths of the edges constructed by the algorithm are sufficiently small then the following holds:*

(1) The algorithm has the same behavior, with respect to edge selection, in space and in the projection onto the tangent plane of the sector center $\mathbf{p}$ of w.

(2) The only reason that no edge might be inserted by the algorithm is that the candidate edge χ_c-intersects an already existing edge of G. The probability of this event is low.

(3) The candidate edge always is a γ-edge, with high probability.

Argumentation: The following argumentation requires that all items involved in the tests of the then-case B, the then-case D, and the else-case D fall into an environment $E(\mathbf{p})$ of $\mathbf{p}$ for which the TPEs of the involved points with respect to the tangent plane at $\mathbf{p}$ are small in comparison to the edge length. But as we already know this can be achieved for a sufficiently small environment $E(\mathbf{p})$ of $\mathbf{p}$, and for sufficiently small edge lengths of the manifold. The environment is defined by the δ of Lemma 9.26 which is independent of $\mathbf{p}$, as we also already know. The edge length is defined by a constant factor which also is independent of $\mathbf{p}$. Thus the required condition is satisfied if the lengths of the edges constructed by the algorithm are limited by a sufficiently small global upper bound, as demanded in the observation. In Chapter 9.1 we have discussed the existence of edge-length bounding sampling sets.

For (1):

The two preceding observations state that the projected current manifold does not have more non-γ-edges than the original one, and that self-overlapping does not occur in the projected manifold if the manifold in space is without self-overlapping. Because G does not have non-γ-edges by assumption, thus G' does not have such edges neither. Thus we have the same starting situation for the projection like for the analysis of the flat case of Section 9.3.

In the following we compare the behavior of the then-case B, the then-case D, and the else-case D for the locally flat case which has been described in Section 9.3 with the situation on curved SF-surfaces.

Then-Case B:

The curved case is treated in the orthogonal projection of the relevant part of G onto the plane $H(w)$ of the sector w. Thus we have to compare the situation in the projection on $H(w)$ for the given surface, and in the projection on the tangential plane $H(\mathbf{p})$ for the projected case (Figure 9.15).

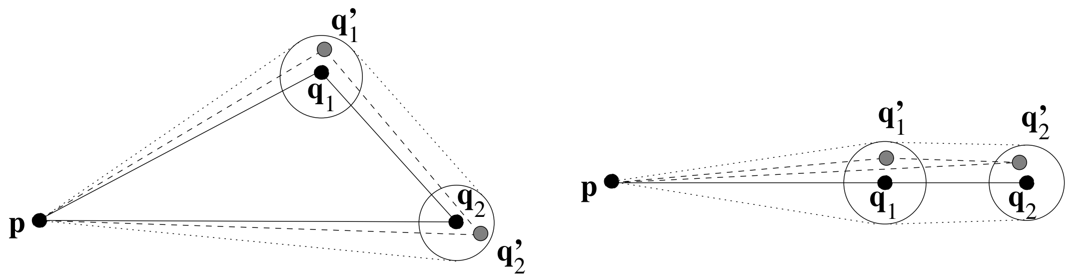


Figure 9.15: Illustration of the argumentation of (1) of Observation 9.32. $t(w)$ is drawn with solid lines, $t(w')$ with dashed lines, and $T(w)$ with dotted lines. The configuration is presented in top- and side-view.

The first question to be asked is whether the candidate sets of the spatial and the projected configuration are the same, up to projection. The candidate set $P_c^F(w)$ consists of all sampling points flat over the sector triangle $t(w)$. We argue that the candidate sets are the same with high probability by showing that

(1) the projections of sampling points flat over $t(w)$ should also be in $t(w')$, and

(2) sampling points which project into $t(w')$ should be flat over $t(w)$, both with high probability.

For the discussion of (1) we use the assumption that all points $\mathbf{s}$ flat over $t(w)$ are in the NN-image of $t(w)$. Aspects of this assumption will be discussed in Section 9.5.2. Under this assumption, the TPE of $\mathbf{s}$ intersects the tangent plane. If the intersection of the TPE with the tangent plane is a subset of $t(w')$, then $\mathbf{s}$ is in $t(w')$. Otherwise $\mathbf{s}$ is close to an edge of $t(w)$. This implies that there are three approximately co-linear points what should occur with only low probability.

For (2) we consider a projected sampling point **s** in $t(w')$. Let $T(w)$ be the convex hull of the TPEs of the three vertices of $t(w)$, cf. Figure 9.15. Because $t(w)$ and $t(w')$ are subsets of $T(w)$, and because the TPE of **s** intersects $T(w)$, **s** is flat over $t(w)$ or close to an edge of $t(w)$. The latter case causes three approximately co-linear points what should occur with only low probability.

Thus (1) and (2) hold, and the two candidate sets can be considered as equivalent, with high probability.

The second question is whether the same point **q** is selected in space and in the projection. According to Section 8.5.2 the algorithm selects **q** as an extremal point in the projection of the candidate points onto the triangle $t(w)$. **q** is a point whose projection has largest distance from the line of edge $\overline{\mathbf{q}_1\mathbf{q}_2}$. We assume that the orthogonal projection $\mathbf{q}'$ of **q** onto the tangent plane does not have this property. We show that under this assumption another candidate point **r** exists so that the line segments $\overline{\mathbf{qr}}$ and $\overline{\mathbf{q}_1\mathbf{q}_2}$ are approximately parallel if the edge lengths of the configuration are small. But the probability of parallelism can be considered as low, so that the probability of the assumption is low, and that the probability is high that the same point **q** is selected in space and in the projection.

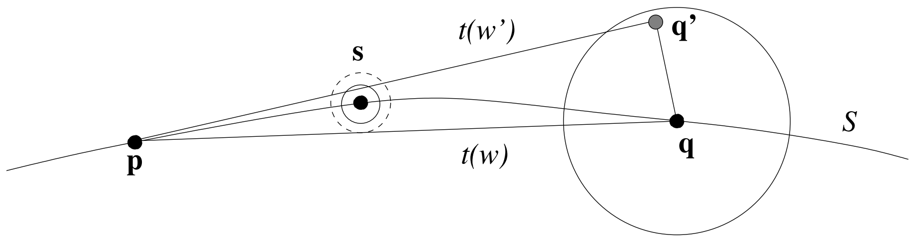


Figure 9.16: Definition of the sector plane environment (SPE) in a 2D representation. The SPE of **s** is drawn as dashed circle, the TPE as solid circle, like the TPE of **q**.

For the candidate points **s** we consider the closed balls centered at **s** with their distance from $t(w)$ as radius (Figure 9.16). We call these balls *sector plane environments* (*SPE*). From the above discussion of (1) we know that the intersections of the TPEs of **s** with the tangent plane are subsets of $t(w')$ with high probability. Because $t(w')$ and $t(w)$ are subsets of $T(w)$, the TPEs of the points **s** intersect $T(w)$, too. Thus the distance of the point **s** from $t(w)$ is at most the radius of the TPE of **s** plus the maximum of the distances of the TPEs of the vertices of $t(w)$. Thus the radius of the SPE of an **s** can be made small, analogous to that of the TPE.

Let **q** be the point selected by the algorithm for the curved case. Thus the projection $\overline{\mathbf{q}}$ of **q** is a point of largest distance from the line of edge $\overline{\mathbf{q}_1\mathbf{q}_2}$. Let **r** be the sampling point whose projection $\mathbf{r}'$ on the tangent plane would be selected by the algorithm for the projection on the tangent plane. $\mathbf{r}'$ is a point of largest distance from the line of edge $\overline{\mathbf{q}_1'\mathbf{q}_2'}$ of $t(w')$. We consider the plane $\overline{H}$ of the points $\overline{\mathbf{q}}_1,\overline{\mathbf{q}}_2,\overline{\mathbf{q}},\overline{\mathbf{r}}$ and the plane H' of the points $\mathbf{q}_1',\mathbf{q}_2',\mathbf{q}',\mathbf{r}'$. If $\mathbf{q} \neq \mathbf{r}$ then $\overline{\mathbf{q}}$ is farer from the line of $\overline{\mathbf{q}_1}$ and $\overline{\mathbf{q}_2}$ than $\overline{\mathbf{r}}$ in $\overline{H}$, and $\mathbf{r}'$ is farer from the line of $\mathbf{q}_1'$ and $\mathbf{q}_2'$ than $\mathbf{q}'$ in H', and we consider an arbitrary continuous motion of the plane $\overline{H}$ to the plane H' so that the plane permanently intersects curves connecting the corresponding points $\overline{\mathbf{q}}/\mathbf{q}'$, $\overline{\mathbf{r}}/\mathbf{r}'$, $\overline{\mathbf{q}_1}/\mathbf{q}_1'$, and $\overline{\mathbf{q}_2}/\mathbf{q}_2'$ which deviate from the straight-line connection just by a small constant multiple of the maximum radii of the TPEs and SPEs. The intersection of the moving plane with the line segments induces vertices $\mathbf{q}_1^*,\mathbf{q}_2^*,\mathbf{q}^*,\mathbf{r}^*$. For at least one location, $\mathbf{q}^*$ and $\mathbf{r}^*$ have equal distance to the line of $\mathbf{q}_1^*$ and $\mathbf{q}_2^*$. Because of the condition of deviation on the curves, the line segments $\overline{\mathbf{q}_1\mathbf{q}_2}$ and $\overline{\mathbf{qr}}$ are approximately parallel, if the TPEs and SPEs are sufficiently small, what can be expected for sufficiently small edge lengths.

For the proof of existence of the desired curves we consider the line of intersection of the planes $\overline{H}$ and H'. This line partitions the planes into half-planes. If the points on every plane all fall into the same half-plane, we choose the straight-line connection between corresponding points as curve (Figure 9.17, left). Otherwise we take the straight-line connections as connecting curves only if they

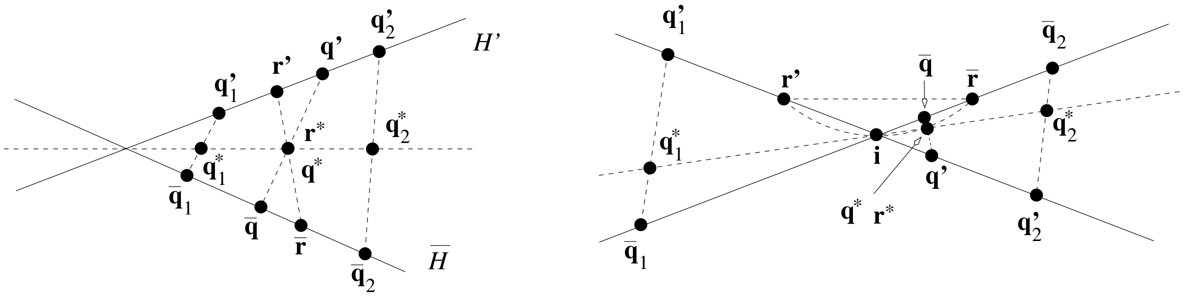


Figure 9.17: Construction of a configuration of points $\mathbf{q}_1^*, \mathbf{q}_2^*, \mathbf{r}^*, \mathbf{q}^*$ in which $\mathbf{r}^*$ and $\mathbf{q}^*$ reach a desired location. In the left case the pairs of points are connected by straight lines. In the right case $\overline{\mathbf{r}}$ and $\mathbf{r}'$ are connected by a circular arc which has a point in common with the intersection of $\overline{H}$ and H'.

traverse the wedge with the smaller angle (Figure 9.17, right). For the other pairs of points, for instance $\overline{\mathbf{r}}$ and $\mathbf{r}'$ in the figure, we take a point $\mathbf{i}$ on the intersection line of $\overline{H}$ and H' so that the triangle $\triangle(\overline{\mathbf{r}}, \mathbf{r}', \mathbf{i})$ maximizes the angle at $\mathbf{i}$ among all those triangles. We take the circular arc defined by $\overline{\mathbf{r}}$, $\mathbf{r}'$, $\mathbf{i}$ as connecting curve. Because the angle of the triangle at $\mathbf{i}$ is at least $90°$, the length of the circular arc is bounded by a constant multiple of the length of the line segment between $\overline{\mathbf{r}}$ and $\mathbf{r}'$. Because the length of this line segment is bounded by a constant multiple of the maximum of the radii of the TPEs and SPEs, and because the circular arc is completely in the two wedges of smaller angle, the connecting curve fulfills these requirements.

Then-Case D:

The first question to be treated here is whether the points of the candidate sets $P_{c,+}^{\beta_c}$ and $P_{c,-}^{\beta_c}$ defined in Section 8.7.1 correspond to the same original points in the original curved case and in the flat projected case. We argue that this holds with high probability.

By definition of the candidate sets, the answer is positive if

(1) the sets of β_c-close points are the same in both cases,

(2) the points project onto the same side with respect to the line through e_3 on the sector plane $H(w)$ in both cases, and

(3) the points $\mathbf{q}$ causing a fold-over are the same in both cases,

with high probability.

If (1) would not hold, then a β_c-close point would exist in one of the cases which is not β_c-close in the other one. β_c-closeness in the projection refers to projected points. Analogously to the argumentation of Observation 9.30 which also works for other environments than the $\beta(\gamma)$-environment used there, it can be shown that this is impossible with high probability.

For the argumentation for (2), we distinguish between two cases (Figure 9.18). The first case is that a point $\mathbf{q}$ exists which is in $P_{c,-}^{\beta_c}$ for the original case, but whose projection $\mathbf{q}'$ is not in $P_{c,-}^{\beta_c}$ for the projected case. Under this assumption, the TPE of $\mathbf{q}$ intersects the tangent plane within $T(w)$. Thus the intersection of the TPE with the tangent plane is a subset of $t(w')$ or $\mathbf{q}$ is close to an edge of $t(w)$. Because the first case is not possible for the assumed relation $\mathbf{q}' \notin P_{c,-}^{\beta_c}$, the second case holds. This implies that there are three approximately co-linear points what should occur with only low probability. Thus the assumption is impossible with high probability.

The second case is that a point $\mathbf{q}$ exists whose projection $\mathbf{q}'$ is not in $P_{c,+}^{\beta_c}$ for the projected case, but $\mathbf{q}$ is in $P_{c,+}^{\beta_c}$ for the original case. Because $P_{c,-}^{\beta_c}$ is empty in the locally flat case, $\mathbf{q}'$ has to be in $P_c^F(w')$,

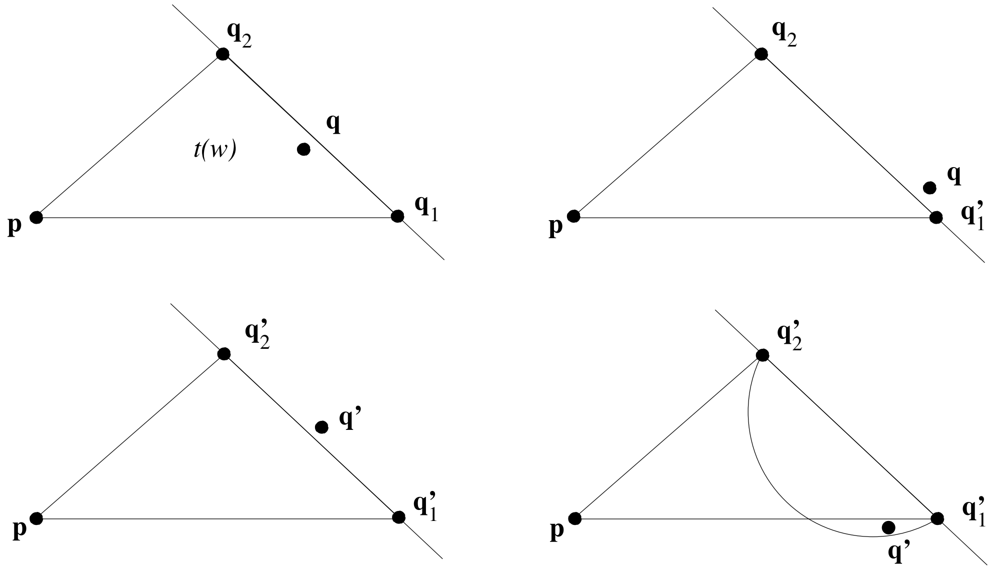


Figure 9.18: Illustration of the argumentation of Observation (1) of the then-case D in Observation 9.32. The left configuration shows the case that a point **q** exists which is in $P^{\beta_c}_{c,-}$ for the original case, but whose projection $\mathbf{q}'$ is not in $P^{\beta_c}_{c,-}$. The right configuration represents the case of a point **q** whose projection is not in $P^{\beta_c}_{c,+}$, but **q** is in $P^{\beta_c}_{c,+}$ for the original case. Additionally, $\mathbf{q}'$ projects outside $t(w)$.

w' the projected sector, or is on the same side of the line through the projected edge e_3, but not in $t(w')$.

If the former holds, because $t(w)$ and $t(w')$ are subsets of $T(w)$, and because the TPE of **q** intersects $T(w)$, **q** is flat over $t(w)$ or close to an edge of $t(w)$. But the first alternative is a contradiction to the assumption of this case. The second alternative leads to three co-linear points whose occurrence should only have low probability.

If the latter holds then $\mathbf{q}'$ is in the difference of $E_{\beta(\gamma)}(e_3')$ minus $t(w')$. The area of this region is small so that it is not very likely to find $\mathbf{q}'$ there. If $\mathbf{q}'$ nevertheless is in this region then the distance to the plane of w should be small, in the order of the radius of the TPEs of the vertices of w. Thus the projection $\overline{\mathbf{q}}$ on this plane can have a distance from the line of e_3 in the order of the TPEs of the vertices of w. But this additionally means that **q** is close to the line of e_3, with high probability. This causes three approximately co-linear points what should occur with only low probability.

The argumentation for (3) is the following. If the projection $\mathbf{q}'$ of a point **q** causes a fold-over in the projected case but not in the original case, or vice versa, then **q** must be close to the line through $\mathbf{q}_1$ and $\mathbf{q}_2$. But this event should happen only with low probability.

The second question is whether the result of point selection is the same in the curved and the projected flat case. In the following we argue that this holds with high probability.

From the discussion of the candidate sets we can conclude that $P^{\beta_c}_{c,-}$ is empty with high probability because this set is empty in the flat case. This means that step 2, inside case, of Section 8.7.2 is never executed, with high probability.

For search step 1, the candidate point $\mathbf{q} \in \overline{P}^{\beta_c}_{c,+}$ selected by the algorithm in the locally flat case is one which encloses the largest angle with e_3 (Section 9.3.2). We give arguments that the same holds in the curved case, with high probability. Since we assume surfaces without edges and ridges, we again set $\beta_c = \beta(\gamma)$, as in Section 9.3.2. Under this assumption, the candidate points are γ-close to e_3. The candidate point **q** initially found by the algorithm in search step 1 has the largest angle. If **q** is γ-close to e_3, the probability that **q** causes a fold-over is low. If a fold-over would occur in space but not in the corresponding projection, then it can be seen that $\mathbf{q}$, $\mathbf{q}_1$, and $\mathbf{q}_2$ are approximately co-linear (Figure 9.3). But the probability of this event should be low. Because we already know from Section 9.3.2 that the probability of a fold-over in the locally flat case is low, the probability that $\mathbf{q}$ causes a fold-over in the curved case is low, too. Because the probability of a $\mathbf{q}$-induced fold-over is low, $\mathbf{q}$ is not replaced with any other sampling point. Thus the candidate point $\mathbf{q}$ selected in search step 1 has indeed the largest angle with e_3.

In the projected case we know from the locally flat case that a projected sampling point with largest angle is selected. Let $\tilde{\mathbf{q}}$ be its corresponding original point. If both points $\mathbf{q}$ and $\tilde{\mathbf{q}}$ would be different, it can be seen by using Corollary 9.28 that $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}$, and $\tilde{\mathbf{q}}$ are approximately on a common circle. The probability of this event should be low. Thus in both cases the algorithm selects the same original point with high probability.

Search step 2, outside case, is entered with a candidate point $\mathbf{q}$ for the original case whose projection $\mathbf{q}'$ is also the candidate point for the projected case. For search step 2, outside case, we have recognized in Section 9.3.2. for the locally flat case and $\mathbf{q}$ γ-close to e_3 that $e = \overline{\mathbf{pq}}$ is intersection-free, and that this observation implies that a replacement of $\mathbf{q}$ does not take place. In order to compare the original case with the flat projected case we give arguments that the following observations hold with high probability:

(1) $e = \overline{\mathbf{pq}}$ is χ_c-intersection-free if and only if its projection $e' = \overline{\mathbf{p}'\mathbf{q}'}$ is intersection free in the flat projection.

(2) For all sampling points $\mathbf{r}$ with $\overline{\mathbf{q}_1\mathbf{r}} \in G$ the relation of the sizes of the angles between $\overline{\mathbf{q}_1'\mathbf{q}_2'}$ and $\overline{\mathbf{q}_1'\mathbf{q}'}$ and between $\overline{\mathbf{q}_1'\mathbf{q}_2'}$ and $\overline{\mathbf{q}_1'\mathbf{r}'}$, where $'$ indicates the projection of the corresponding points, is the same as for the sizes of these angles between the original line segments.

(1) is an immediate consequence of Observation 9.31.

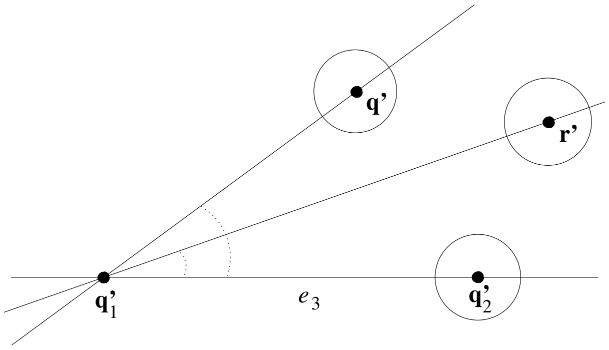


Figure 9.19: Illustration of the tolerances on angles between lines imposed by the radius of the TPEs of the points $\mathbf{q}_2$,$\mathbf{r}$, and $\mathbf{q}$. A perturbation of the locations of the points within the TPEs does not change the relation of the sizes of the angles of interest.

For (2) we consider the cases of Figure 9.7. With high probability $\mathbf{q}'$ and $\mathbf{r}'$ have a distance larger than two times the maximum of the radii of the TPEs of $\mathbf{q}_2'$ and $\mathbf{q}'/\mathbf{r}'$ from the line of e_3'. Otherwise the vertices of e_3 and $\mathbf{q}/\mathbf{r}$ would be approximately co-linear what should be an event of low probability. We assume a configuration of this type (Figure 9.19). If $\mathbf{r}'$ has a distance larger than twice the maximum of the radii of the TPEs of $\mathbf{r}'$ and $\mathbf{q}'$ from the line through $\mathbf{q}_1'$ and $\mathbf{q}'$, and $\mathbf{q}'$ has a distance larger than twice the maximum of the radii of the TPEs of $\mathbf{r}'$ and $\mathbf{q}'$ from the line through $\mathbf{q}_1'$ and $\mathbf{r}'$, then the angles have the same relation in the projected and in the original case. The reason is that perturbation within an environment of the size of a TPE is not sufficient to change the relation. The worst case is that the points are moved within the projection plane. Thus, if the relation changes, $\mathbf{r}$ is close to the line of $\overline{\mathbf{q}_1\mathbf{q}}$, or $\mathbf{q}$ is close to the line of $\overline{\mathbf{q}_1\mathbf{r}}$. In both cases we have three approximately co-linear vertices. But this event has low probability. Thus the probability that the relation of the two angles is the same in both cases is high.

Else-Case D:

The only action of the algorithm is to insert e_3 into G if e_3 does not intersect an edge of G. But by Observation 9.31 the status of e_3 in the original case and the projection e_3' of e_3 in the projected case is the same with respect to χ_c-intersection. Thus the algorithm has the same behavior in both cases.

For (2):

From Observation 9.25 we know that in the locally flat case the event that the candidate edge intersects an already existing edge of G does not occur. Because the projected case is locally flat, the candidate edge of the projection does not intersect an already existing edge. Because for both cases the algorithm has the same behavior, the candidate edge of the original case is the corresponding edge of the projected edge. By Observation 9.31 this original edge has the same behavior with respect to χ_c-intersection for the original case like the projected edge with respect to intersection for the projected case, with high probability. Thus the candidate edge of the original case is χ_c-intersection-free with high probability.

For (3):

From Observation 9.25 we know that in the locally flat case the candidate edge always is a γ-edge, with high probability. Because the projected case is locally flat, the candidate edge of the projection is a γ-edge with high probability. Because for both cases the algorithm has the same behavior, the candidate edge of the original case is the corresponding edge of the projected edge. By Observation 9.30 this original edge has the same behavior with respect to the γ-edge property for the original case like the projected edge with respect to intersection for the projected case, with high probability. Thus the candidate edge of the original case is a γ-edge with high probability. ∎

9.5 NN-Embeddability

NN-embeddability of a triangular manifold M into a surface S means that the nearest-neighbor image of M maps one-to-one to S under the nearest-neighbor assignment. In the following we present observations which show that the M constructed by the algorithm should have this property with high probability if the sample set is chosen so that the maximum angle of the triangles is bounded by γ, the edges are sufficiently short, and the dihedral angles between neighboring triangles are sufficiently large. We know from previous sections that sample sets with this property exist.

9.5.1 General Observation

The purpose of this section is to give arguments for the following observation which also summarizes the investigations performed up to now.

Observation 9.33 (Reconstruction property of the triangulation algorithm) *Let S be an SF-surface. Then a sampling set P exists so that with high probability the triangular mesh M delivered by the triangulation algorithm is NN-embeddable, provided that for all sectors w occurring during execution of the algorithm the candidate sets $P_c^F(w)$ and $P_c^{\beta_c}$ comprehend with high probability all points of P which have an NN-image falling into the triangle $t(w)$.*

Argumentation: We distinguish between two cases, local NN-embeddability and global NN-embeddability.

Local NN-embeddability denotes the NN-embeddability of every sub-mesh consisting of a pair of triangles sharing a common edge. Local NN-embeddability should be guaranteed by the strategy of the triangulation algorithm

(1) to generate γ-edges and thus triangles with maximum angles less than γ preferably,

(2) to select sectors with large dihedral angles with their neighboring sectors in their umbrella preferably (Section 8.3).

Triangles satisfying (1) are NN-embeddable if their edge length is sufficiently small (Theorem 6.12). By (2), if the δ_c-bound of the algorithm on the dihedral angle of two adjacent triangles is sufficiently large, and if one of two adjacent triangles is known to be oriented close to the normal vectors of the surface, then the other should be, too, and the NN-images of the pair of triangles should be disjoint. This holds if both triangles are NN-embeddable and have sufficiently short edges (Theorem 6.16).

For sample sets with the Samp_2-property, the edge length can be held small with high probability (Observations 9.6 and 9.32). Thus local NN-embeddability should be achieved with high probability by the algorithm.

Global NN-embeddability means that the NN-images of any two triangles do not intersect. Global NN-embeddability should be guaranteed by the strategies of the triangulation algorithm

(1) to treat the points over a triangle in case B,

(2) to treat intersecting edges by

   (a) construction in case B of the algorithm,

   (b) explicit χ_c-intersection tests in the other cases of edge insertion.

Two non-adjacent triangles may NN-intersect if a vertex of one of the triangles is in the interior of the other one, or if the two triangles have NN-intersecting edges. Both cases are covered by (1) and (2).

The complete treatment of points over a triangle in (1) is stated as a condition of this observation. More on this subject is told in Section 9.5.2.

The χ_c-intersection tests of (2) should yield the correct result with respect to NN-intersection with high probability, cf. Theorem 7.23 and the subsequent argumentation. ∎

9.5.2 A Candidate Environment Covering the NN-Image of a Triangle

In Section 8.5.1 the candidate environment of flat sample points over a triangle has been defined as an intuitive type of candidate environment. This definition is sufficient in the flat case because exactly the sample points in the interior of a triangle are covered by it. The sample points falling on an edge of the triangle are in an edge environment and are treated in the then-case D.

The candidate environment of flat points over a triangle also works well in practice for the curved case. However, we do not have a proof that this environment, together with the candidate environments of its edges, comprehends the NN-image of the triangle of the surface. In the following a type of candidate region is defined for which we can prove this property for SF-surfaces. The environment can possibly contain some more points, but we can show that they are in a connected environment of the NN-image of the triangle. Thus this type of candidate sets might be a good starting point for finding the sample points in the NN-image of the triangle reliably or at least with high probability.

The new candidate region is defined as follows.

Definition 9.34 (NN-candidate environment of a triangle) *Let $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ be a triangle, and $r(t)$ denote the length of the longest edge of t. The **nearest-neighbor (NN) candidate environment** $C^N(t)$ of t is defined as the union of the three balls of radius $r(t)$ and centers* **p**, **q**, *and* **r**.
*For a finite set P of sample points of a surface S, $P(t) = P \cap C^N(t)$ is denoted as **NN-candidate set** of t.*

In the following we will show that for sufficiently small triangles $P(t)$ does indeed contain all relevant sample points.

Lemma 9.35 *Let $r(t)$ be the length of the longest edge of a triangle t. Then the following holds:*

(1) Every point of t is in distance less or equal to $\frac{\sqrt{2}}{2}r(t)$ from one of the vertices of t.

(2) The NN-candidate environment $C^N(t)$ contains in particular all those points in space which are in distance less or equal to $\frac{\sqrt{2}}{2}r(t)$ from t.

Proof: Every point of t is in distance less or equal to $\frac{\sqrt{2}}{2}r(t)$ from one of the vertices of t. The reason is that every point of t has a distance of at most $\frac{1}{2}r(t)$ from one of the edges of t. This can be seen by considering the three rectangular quadrilaterals, each of which shares an edge with t and the opposite edge on the bisector between that edge and the opposite vertex of t, cf. Figure 9.20. The three quadrilaterals cover t. Hence the distance from one of the vertices is at most $\sqrt{(\frac{1}{2}r(t))^2 + (\frac{1}{2}r(t))^2} = \frac{\sqrt{2}}{2}r(t)$.

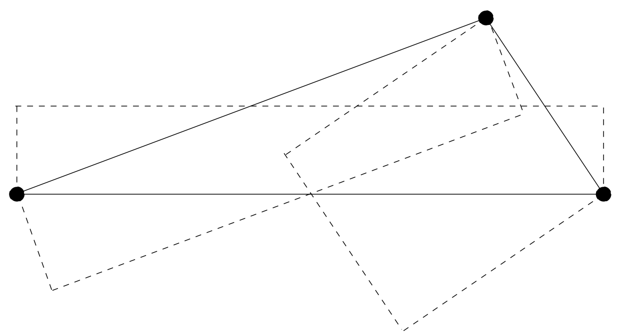


Figure 9.20: The three quadrilaterals (dashed lines) that cover the triangle t.

Let **q** be a point in space of distance less or equal to $\frac{\sqrt{2}}{2}r(t)$ to t, and $\mathbf{q}'$ its closest point on t. If $\mathbf{q}'$ is a vertex of t, $d(\mathbf{q}, \mathbf{q}') \leq \frac{\sqrt{2}}{2}r(t)$ is less than $r(t)$, and thus **q** is in the ball of radius $r(t)$ around $\mathbf{q}'$. If $\mathbf{q}'$ is on an edge or in the inner of the triangle, $\overline{\mathbf{q}\mathbf{q}'}$ is perpendicular to the edge or to the triangle. In both cases, a line perpendicular to $\mathbf{q}'$ intersects the ball of a triangle vertex **p** from which $\mathbf{q}'$ has a distance $d(\mathbf{q}', \mathbf{p}) \leq \frac{\sqrt{2}}{2}r(t)$ in a point $\mathbf{q}''$ which has distance $d(\mathbf{q}'', \mathbf{q}') = \sqrt{r(t)^2 - d(\mathbf{q}', \mathbf{p})^2} \geq \sqrt{r(t)^2 - \frac{1}{2}r(t)^2} = \frac{\sqrt{2}}{2}r(t)$ from $\mathbf{q}'$. Thus for $d(\mathbf{q}, \mathbf{q}') \leq \frac{\sqrt{2}}{2}r(t)$, we get $d(\mathbf{q}'', \mathbf{q}') \geq d(\mathbf{q}, \mathbf{q}')$, and thus **q** is in the closed ball around **p** with radius $r(t)$. ■

Theorem 9.36 (Candidate environment containment of NN-embedded triangles) *Let S be a surface and $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ be a triangle with $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$. Then the NN-image $\mathbf{f}(t)$ is a subset of $C^N(t)$.*

Proof: Let $\mathbf{f}(\mathbf{p})$ be the unique nearest neighbor on S of a point $\mathbf{p} \in t$. Then $d(\mathbf{p}, \mathbf{f}(\mathbf{p})) \leq \frac{\sqrt{2}}{2}r(t)$, because of (1) in the preceding lemma, and because the vertices of t are on S. Let $\mathbf{p}'$ be a point of t with closest distance of $\mathbf{f}(\mathbf{p})$ to t. Then $d(\mathbf{p}', \mathbf{f}(\mathbf{p})) \leq d(\mathbf{p}, \mathbf{f}(\mathbf{p})) \leq \frac{\sqrt{2}}{2} r(t)$. By (2) of the lemma that means that $\mathbf{f}(\mathbf{p}) \in C^N(t)$. ■

Theorem 9.37 (SF-fringe containment of candidate environments) *Let S be a compact SF-surface without boundary with SF-radius r. Let t be a triangle with $r(t) < r$. Then $C^N(t)$ is a subset of the SF-fringe of radius r of S.*

Proof: $C^N(t)$ is the union of three balls of radius $r(t)$ and centers on S. Thus every point in $C^N(t)$ has a distance of at most $r(t)$ from S, and thus, by definition of the r-fringe, is a subset of the fringe. ■

Theorem 9.38 (Connected sub-surfaces) *Let S be a compact SF-surface without boundary with an SF-radius r. Let $B_{r/2}(\mathbf{p})$ be the ball of radius $\frac{r}{2}$ centered at a point $\mathbf{p} \in S$. Then $S(\mathbf{p}) := B_{r/2}(\mathbf{p}) \cap S$ is a connected subsurface of S.*

Proof: We consider the connected component of $S(\mathbf{p})$ which contains $\mathbf{p}$. This connected component splits $B_{r/2}(\mathbf{p})$ into two regions so that for each of the tangent balls $B_1(\mathbf{p}), B_2(\mathbf{p})$, the intersections $B_1(\mathbf{p}) \cap B_{r/2}(\mathbf{p})$ and $B_2(\mathbf{p}) \cap B_{r/2}(\mathbf{p})$ are in different regions. That can be seen by inspecting the intersection of the normal planes at $\mathbf{p}$ with the surface and the balls $B_1(\mathbf{p})$ and $B_2(\mathbf{p})$. In each of these planes, the connected component is a curve through $\mathbf{p}$ which induces two regions each of which contains one of the discs induced by the balls $B_1(\mathbf{p}), B_2(\mathbf{p})$. Otherwise a boundary point of the curve, and hence of S, would exist. This transfers to the separation in space, because of the continuity of S.

Now let $\mathbf{q}$ be a point in $B_{r/2}(\mathbf{p})$ which is not on the considered connected component. We will show that $\mathbf{q}$ is not on S. But that means that $S(\mathbf{p})$ is just the connected component of $\mathbf{p}$ what proves the theorem.

For $\mathbf{q}$ in $B_1(\mathbf{p})$ or $B_2(\mathbf{p})$, $\mathbf{q}$ is not on S. Thus let us focus on the other case. We consider the closest points $\mathbf{q}_1'$ and $\mathbf{q}_2'$ of $\mathbf{q}$ on the balls $B_1(\mathbf{p})$ and $B_2(\mathbf{p})$, respectively. From the geometric arrangement it can be seen that the distance $d(\mathbf{q}, \mathbf{q}_i') \leq \frac{r}{2}$, $i = 1, 2$. Furthermore, one of the line segments $\overline{\mathbf{q}\mathbf{q}_i'}$, $i = 1, 2$, intersects the surface in a point $\mathbf{q}'$. Thus the distance of $\mathbf{q}$ from the surface is at most $\frac{r}{2}$. But that means that $\mathbf{q}$ is in one of the tangent balls of radius r at $\mathbf{q}'$, and thus $\mathbf{q}$ cannot be on S. ■

Corollary 9.39 (Connectedness of the subsurface in the candidate environment) *Let S be a compact SF-surface without boundary with an SF-radius r. Let $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ be a triangle with the length of the longest edge $r(t) < \frac{r}{2}$. Then $S' := C^N(t) \cap S$ is a connected subsurface of S.*

Proof: By the preceding theorem, $S(\mathbf{p})$, $S(\mathbf{q})$, and $S(\mathbf{r})$ are connected surfaces. Because $\mathbf{p}, \mathbf{q}$, and $\mathbf{r}$ are contained in all of those three surfaces, the union of them which is just S' is connected, too. ■

Theorem 9.40 (r-fringe containment of a triangle) *Let S be a compact SF-surface without boundary with an SF-radius r and $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ be a triangle with $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$. If $r(t) \leq \sqrt{2}r$, then t is in the r-fringe of S.*

Proof: As we know, the maximum distance of a point of t from S is at most $\frac{\sqrt{2}}{2} r(t)$. Because $\frac{\sqrt{2}}{2} r(t) \leq r$, all points of t have a distance of at most r from S. Thus, by definition of the r-fringe, t is in the r-fringe of S. ■

Based on these observations, the following corollary formulates the usefulness of the candidate environment $C^N(t)$.

Corollary 9.41 *Let S be a compact SF-surface without boundary with an SF-radius r, P be a finite set of sample points on S, and $t = \triangle(\mathbf{p}, \mathbf{q}, \mathbf{r})$ be a triangle with $\mathbf{p}, \mathbf{q}, \mathbf{r} \in S$ with $r(t) < \frac{r}{2}$. Then*

(1) $C^N(t)$ *contains all sample points in the NN-image* $\mathbf{f}(t)$ *of* t.

(2) $C^N(t) \cap S$ *is connected, that is only sample points of a connected environment around* $\mathbf{f}(t)$ *are considered.*

Proof: The assertion (1) is identical to that of Theorem 9.36 about candidate environment containment of NN-embedded triangles. (2) is the assertion of Corollary 9.39 about connectedness of the subsurface in the candidate environment. ■

9.6 Boundaries

The boundary of a triangular manifold is given by the edges to which just one triangle is incident. Boundary edges are generated by the reconstruction algorithm if a sector is not closed to a triangle. Let us again consider the situation in the plane.

In the plane, the boundary consists of closed polygonal chains which also may have tree-like branches and which bound a region outside the triangulation. It is well known, that for every polygonal region at least one sector defined by consecutive edges $e_1 = \overline{\mathbf{pq}_1}$, $e_2 = \overline{\mathbf{pq}_2}$, does exist which can be separated from the region by a chord $e_3 := \overline{\mathbf{q}_1\mathbf{q}_2}$ which is completely in the region, so that the resulting triangle is free of points or edges. The only reason for the algorithm not having inserted e_3 is that the angle at **p** exceeds the boundary control angle γ_c'. That means that all sectors which allow a chord have an angle exceeding γ_c' which usually is chosen large.

If a sector does not allow a chord, its point **p** is either a concave point with respect to the region, or it induces a triangle which contains at least one vertex (the triangulation is assumed to be intersection-free). But in that case, a line segment between **p** and one of those vertices would have been inserted into the graph according to the then-case B. Thus that case does not happen.

In summary, a boundary vertex of a region induced by boundary edges either has a convex large angle, or is concave with respect to the region.

The result of the discussion can be used in two manners. The first interpretation is that it is unlikely that such a polygonal region does occur if the sample points are equally distributed in a surface filling manner. The reason is that the area of a region with the described property is so large that for that case the probability is high that one of the sample points falls into the region.

The second consequence is that the algorithm is capable of identifying intended holes or boundaries if the surface is adequately sampled. An adequate sample locates the sample points on the boundary so that the angles of the sector of the polygonal chain obtained by connecting consecutive sample points exceed γ_c'. For a continuous curve of bounded curvature that can be achieved by sufficiently close sample points. Additionally, the sample points should be denser than the size of the hole, where the size of the hole can be characterized by the diameter of a maximum sphere which can be moved through the hole. The sample points in the interior close to the boundary are set with an analogous density. For that choice, the sectors on the boundary should occur as sectors of the cycle of the boundary vertices of the surface description graph which are not completed to a triangle because of the large angle γ_c'. An edge between the two edges of a boundary sector is unlikely to occur in the surface description graph. The reason is that it is unlikely to occur in the clustered environment graph, nor as a result of edge connection with points in a triangle or edge-close points which are the only possibilities where sectors might be splitted.

Figure 9.21 shows an example of a surface with hole. In the middle, the bound γ_c' has been so small that the hole has been filled, whereas on the right side the hole has been reconstructed by using a suitable large value of γ_c'.

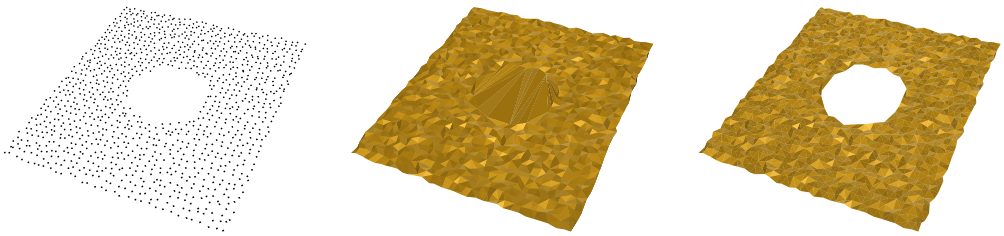


Figure 9.21: The detection of borders in a data site.

9.7 Empirical Investigations

Observation 9.6 states the existence of favorable extensions of a given sample set of a surface, but does not give a concrete algorithm of construction. For that reason we have applied the algorithm to random sample sets taken from a semi-sphere, from the outer and the inner of a torus, and from a flat square (Figure 9.22), with approximately 500 and 1000 points which have been generated as described at the end of Section 7.2.1. Additionally performed measurements with more points show no principle differences, and thus are not presented here.

In all cases $\beta = 1$ has been used for the clustered β-EG which has served as input SDG for the algorithm of triangulation. The parameters γ (for $\beta_c := \beta(\gamma)$) and γ_c' have both been set to $135°$, χ_c was set to $75°$ and the dihedral angle bound to $\delta_c = 60°$. The reconstructions obtained from the sample sets are shown in Figure 9.23.

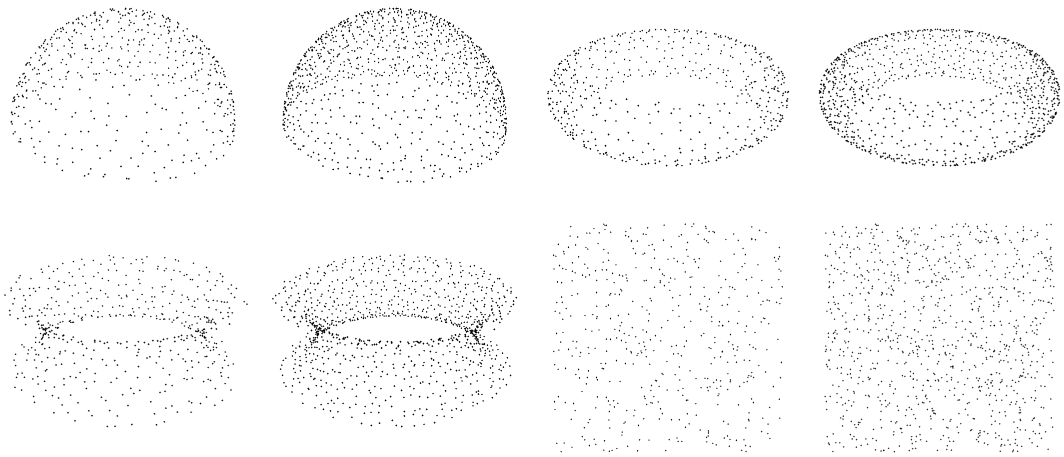


Figure 9.22: The sample sets used for the empirical analysis.

Figures 9.24 to 9.31 compile the data of the measurements. The horizontal axis of every plot can be understood as time axis: i represents the i-th sector treated by the algorithm.

Figure 9.24 shows the size of the sector angles. In the flat case, the sectors are processed according to increasing angle (Definition 9.34). The reason of the scattering effect is that new sectors which may have smaller angles are generated if an edge is inserted into G. An interesting effect is that

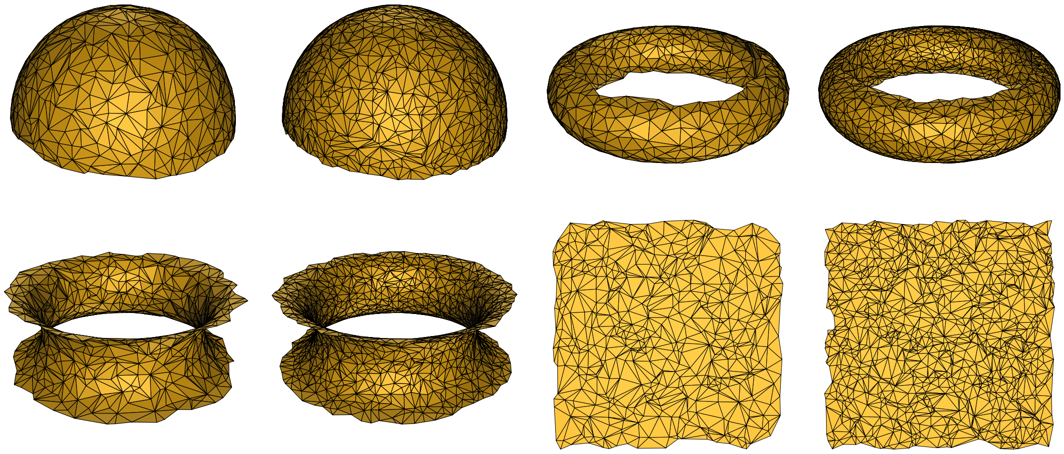


Figure 9.23: The reconstructed surfaces.

nevertheless the tendency is that the angle size increases during execution of the algorithm. The plot shows that the angles are indeed bounded by $135°$.

In the plots of the planar square, a sorting effect can be noticed. The reason is that in that case the sector angles, and not the dihedral angles to neighboring sectors are the sorting criterion. Another observation is that very small angles do not occur frequently.

Figure 9.25 displays the number of occurrences of inserted γ-edges and non-γ-edges, respectively. Almost all of the inserted edges are γ-edges. The number of non-γ-edges is between 0 and 10. This also corresponds to the observation of Section 9.3.2. Moreover, the dihedral angles of adjacent triangles of these non-γ-edges always fit smoothly into the surrounding structure. Most of the dihedral angles between adjacent triangles at these edges exceeded $170°$ in the experiments. The smallest angle ever found has been about $156°$. Obviously, the edge selection process chooses the configurations well, so that a good surface mesh is generated.

Figure 9.26 presents the number of occurrences of the configurations during a reconstruction that a sample point is "over a triangle" (case B, "over trian"), in the $\beta(\gamma)$-environment of edge e_3 (then-case D, "near to edge"), and that e_3 is a γ-edge or an already existing edge (else-case C and case D, "normal"). The case "over a triangle" does practically not occur, the curve is about constant equal to 0. For the overwhelming majority of the sectors edge e_3 is taken and non-γ-e_3-edges do not occur too often. A reason is that for the relatively short edges the probability that a random sample point falls into the $\beta(\gamma)$-environment of the edge should be considerably smaller than it does not, because of the small area of this environment compared to the area of the whole square. The curves of the case of non-γ-edges e_3 (then-case D, "near to edge") increase slightly over-proportionally.

Figure 9.27 shows the number of occurrences of edge types $e_3 = \overline{\mathbf{q}_1\mathbf{q}_2}$ and $e = \overline{\mathbf{pq}}$ over time for the min-max triangulation of the then-case D. Here, the majority of the edges is of type $\overline{\mathbf{pq}}$. This corresponds to the observation of Section 9.3.2.

Figure 9.28 displays the length of the inserted edges. For comparison, the average edge length, the standard deviation, and the maximum edge length of the initial SDG are depicted, too. The distribution of the point density shows that the length of the majority of edges is at most twice of the average length of the SDG, and less than the length of the maximum length of an SDG-edge.

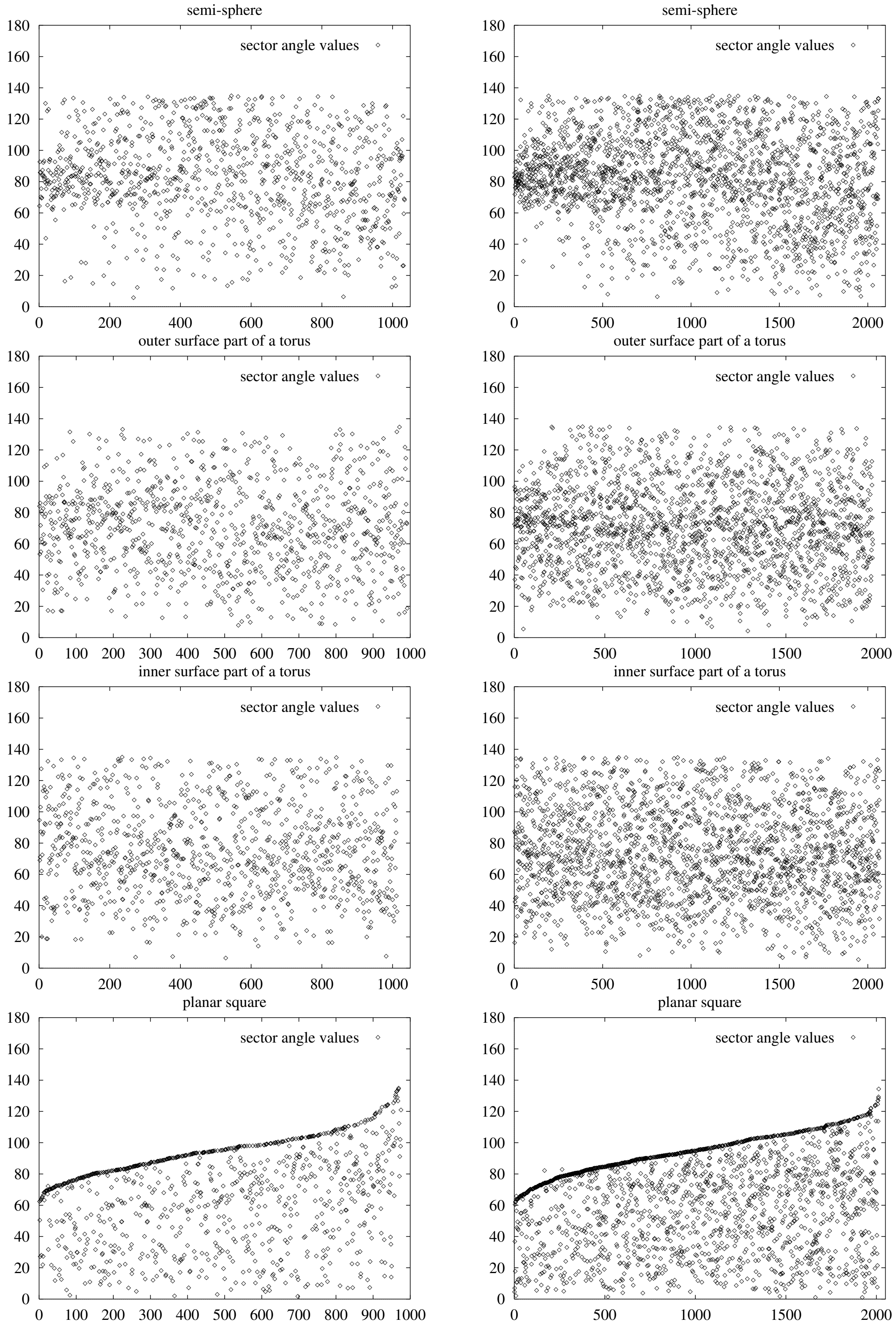


Figure 9.24: The size of the angle of every processed sector.

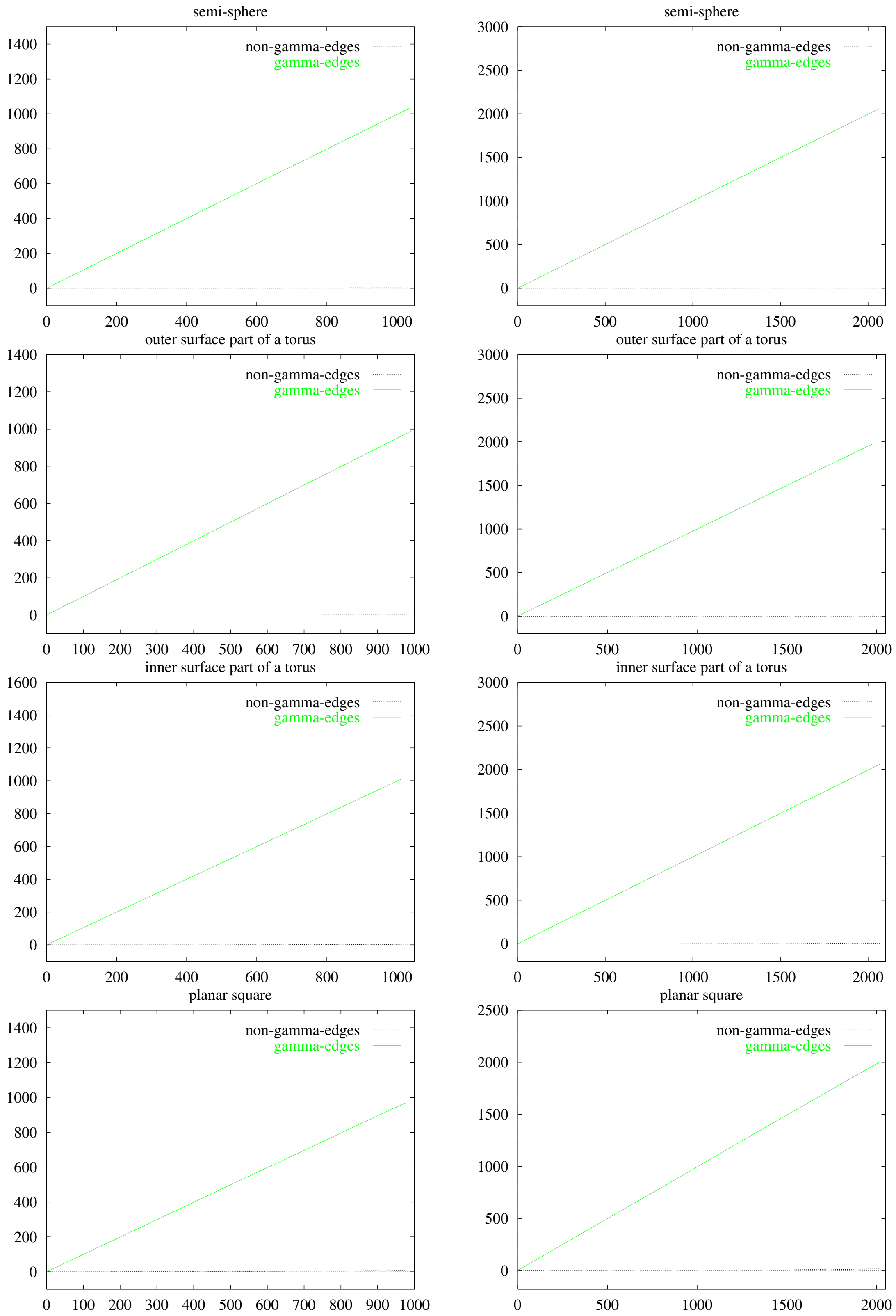


Figure 9.25: The number of occurrences of inserted γ-edges and non-γ-edges, respectively.

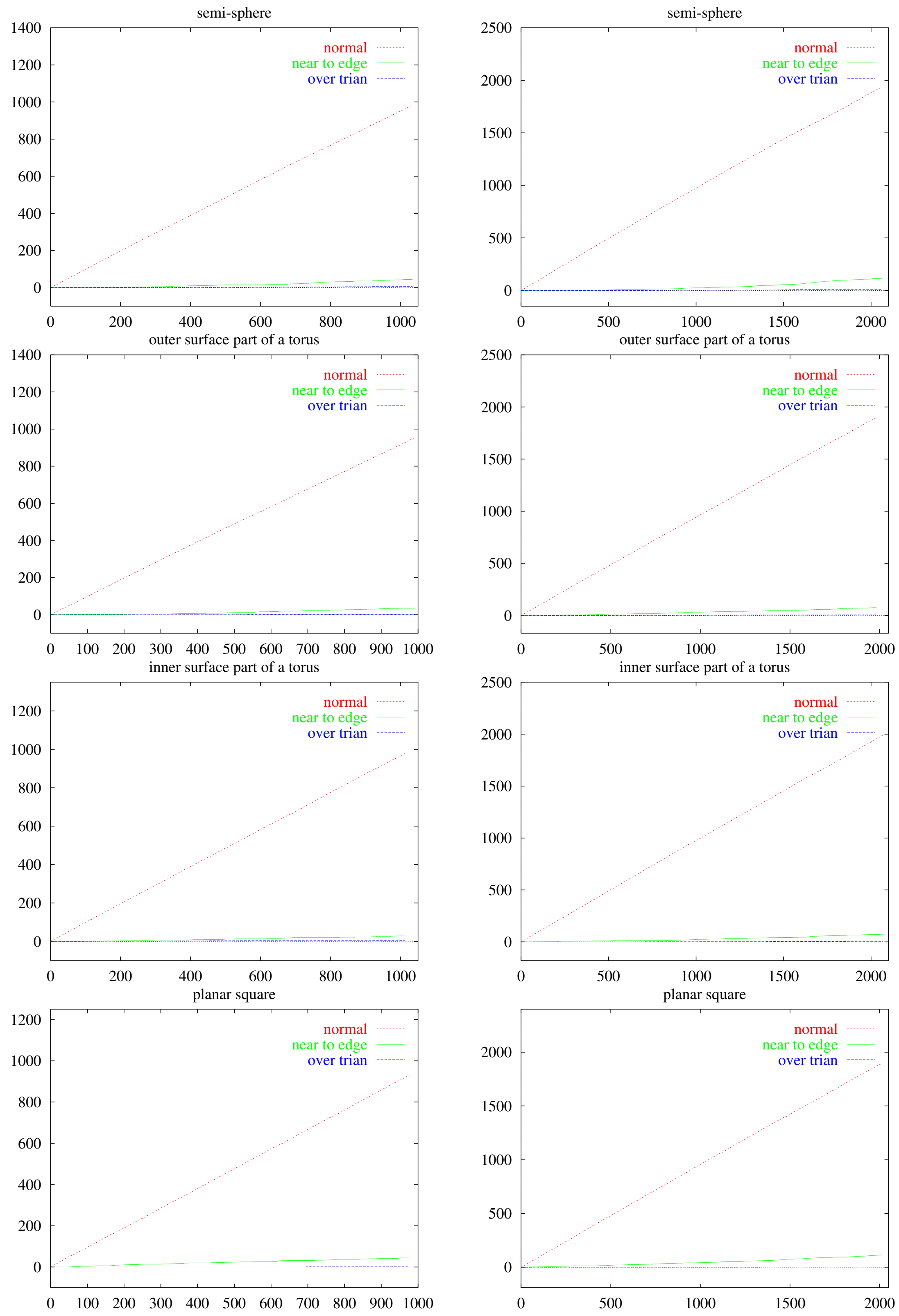


Figure 9.26: The number of occurrences of the configurations "over a triangle" (case B, "over trian"), non-γ-edges e_3 (then-case D, "near to edge"), and γ-edges e_3 or an already existing edge (else-case C and case D, "normal").

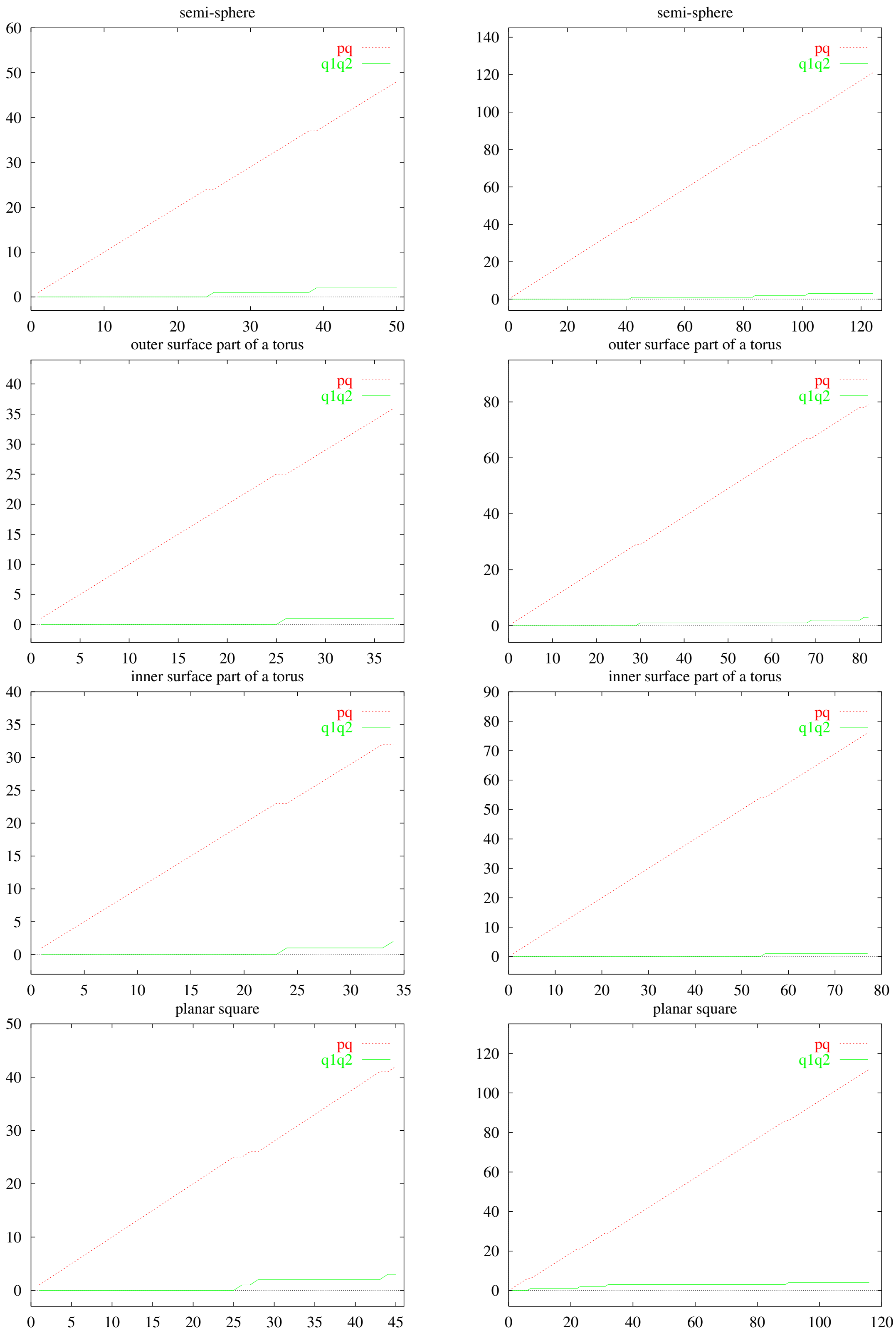


Figure 9.27: The number of occurrences of edge types $e_3 = \overline{\mathbf{q}_1\mathbf{q}_2}$ and $e = \overline{\mathbf{pq}}$ over time for the min-max triangulation of the then-case of step D.

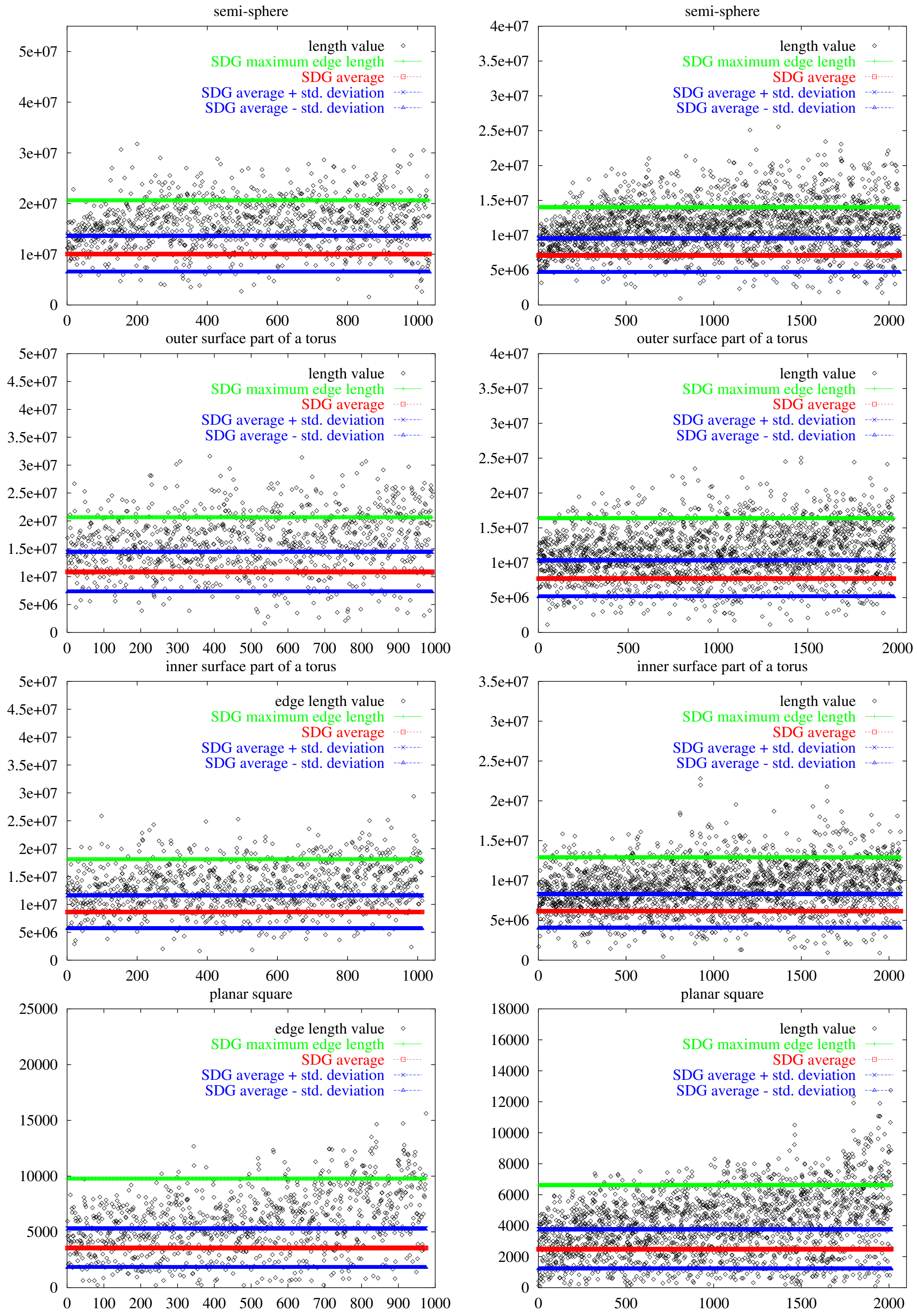


Figure 9.28: The length of the inserted edge: for comparison, the average edge length, the standard deviation, and the maximum edge length of the initial SDG are depicted, too.

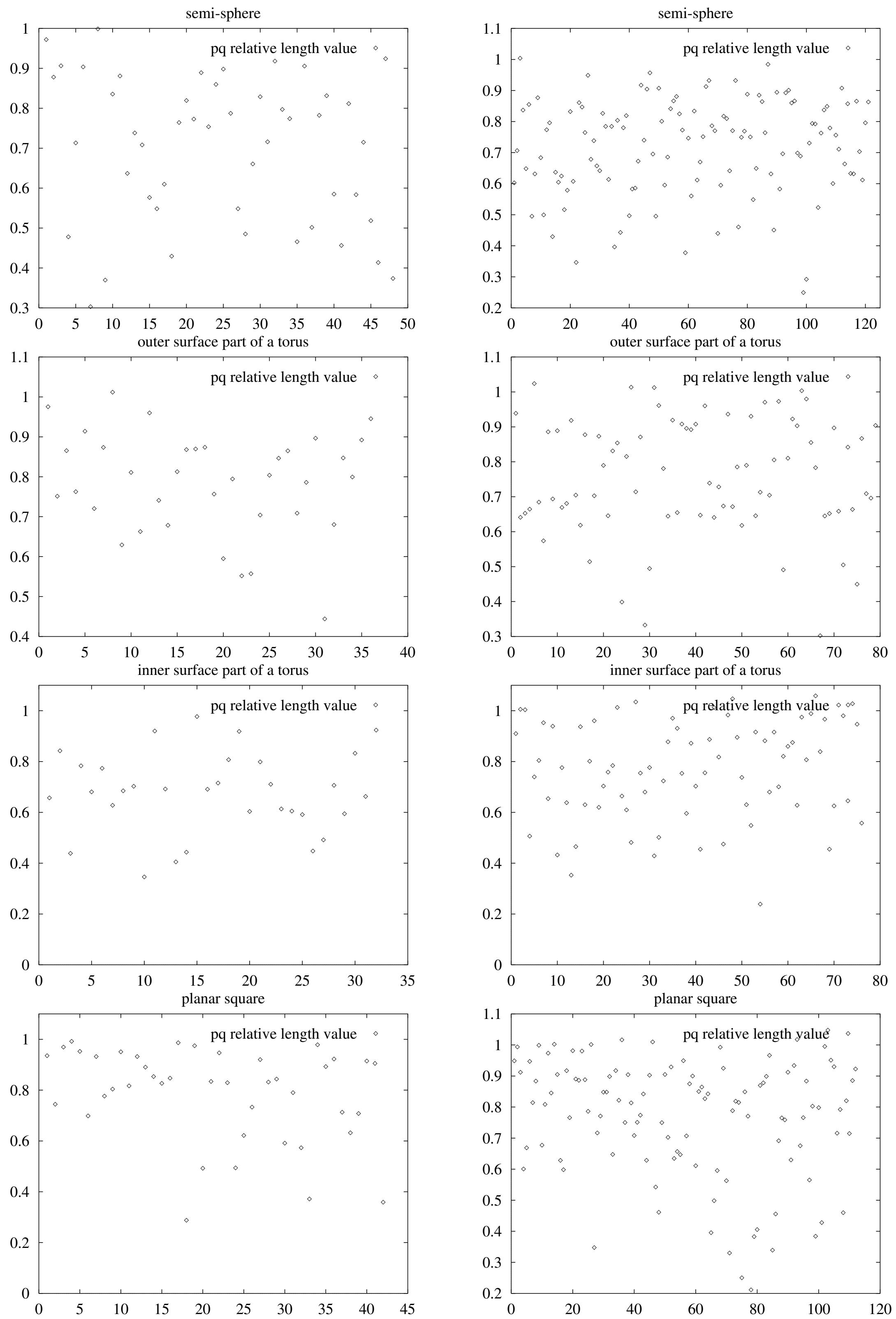


Figure 9.29: The ratio of the length of the newly inserted edge $\overline{\mathbf{pq}}$ and the maximum length of the already existing edges e_1, e_2 of the current sector.

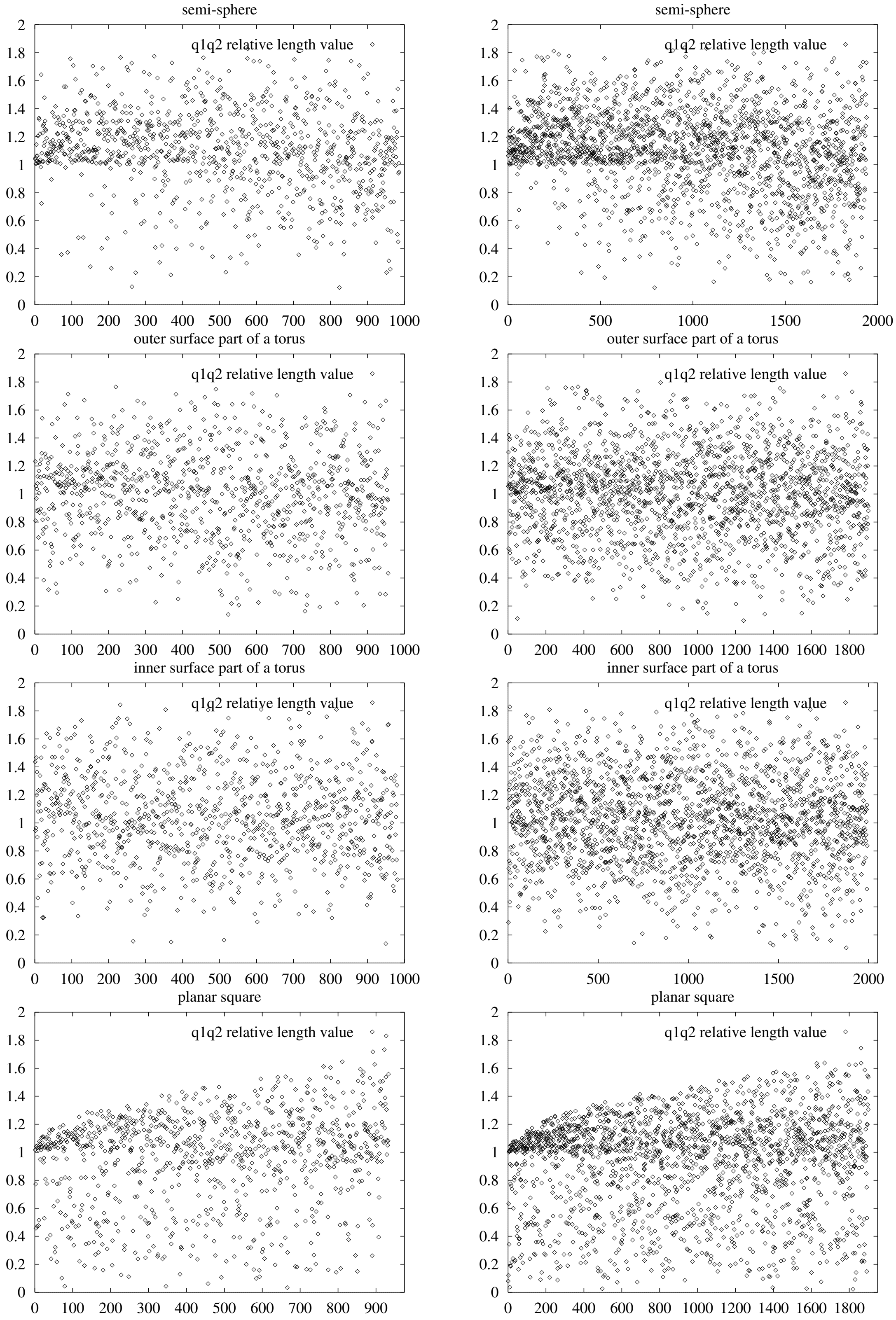


Figure 9.30: The ratio of the length of the newly inserted edge $\overline{\mathbf{q}_1\mathbf{q}_2}$ and the maximum length of the already existing edges e_1, e_2 of the current sector.

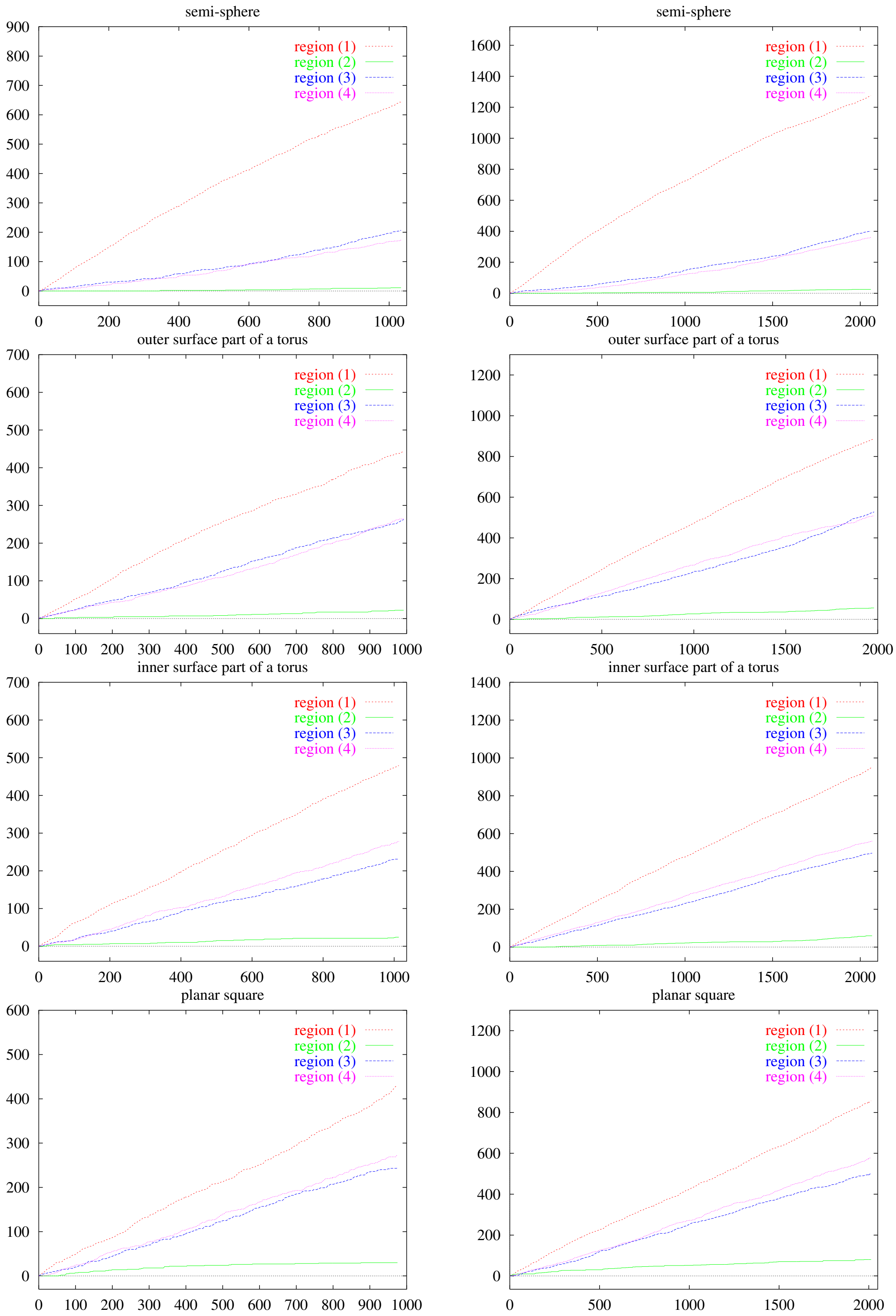


Figure 9.31: The location of point **p** with respect to the regions of Figure 9.1.

Figures 9.29 and 9.30 show the ratio of the length of the newly inserted edge and the maximum length of the already existing edges e_1, e_2 of the current sector. The point density distribution shows that the factor very often is less than 1.2. It increases during the advance of the algorithm, but still many edges fulfill this bound. The plot confirms the trivial upper bound of 2.

The edge length ratios for the planar square seem to be somewhat less than those of the other sample sets, and less scattered. The semi-sphere also does yield less scattered ratios.

Figure 9.31 shows the location of point **p** with respect to the regions of Figure 9.1. In the majority of cases, **p** is located in region 1, and region 2 is used rarely This matches with the observation that the majority of sector angles in Figure 9.24 is larger than $60°$. A somewhat surprising effect is the non-symmetric distributions of the points falling into the symmetric regions 3 and 4.

In summary, the empirical results show that the triangulation algorithm behaves for the random samples as predicted by the analysis. It has turned out that random sampling is already a useful choice for the algorithm. Particularly well suited would be samples arranged on an approximately uniform quadrilateral mesh because in that case the shape of the resulting triangles would be very favorable.

9.8 Reconstruction at Ridges and Sharp Edges

The analysis performed up to now has focused on smooth surfaces. As we can see from the examples of Figures 8.6 and 8.7 the reconstruction algorithm also shows a favorable behavior at ridges and sharp edges. Usually, the precision of reconstruction of the algorithm depends strongly on the surface description graph of the first phase. For clustered environment graphs, its precision can be that of the EMST explained in Figure 4.4 of Chapter 4, that is, in theory the sharpest turn at a ridge can be $60°$. This value, however, is only achieved if the points of the environment have equal distances around a ridge. In practice, the observed precision is at least $90°$ and the average precision is between $60°$ and $90°$. This value directly depends on the length ratio between two incident edges to a point that is on or near to a ridge.

A good choice in order to get a favorable reconstruction of a ridge or a sharp edge is to choose sample points directly on the ridge. The cup shown in Figure 8.6 is an example. By the remarks of the preceding paragraph, this strategy, however, is only reasonable if the surface turn is not too sharp, so that the structure of the environment graph can follow the surface. A further limitation of sample points directly on a ridge is the scanning hardware because data acquisition at really sharp features usually cause physical measuring problems with the consequence of noisy data. These problems often occur for optical scanners.

An other strategy applicable in those cases is to sample just short before and behind a ridge. Then it can be expected that the environment graph follows the sharp surface turn with a ”round” approximation. Figure 4.4 illustrates this idea.

Random sampling at ridges and sharp edges is probably the worst approach because nothing can be told about the position of the points and the resulting structure of the environment graph with respect to the ridge or edge.

9.9 Discussion

The main results of this chapter are summarized in the Observations 9.6, 9.25, 9.32, and 9.33. They demonstrate the existence of sample sets for smooth surfaces for which the algorithm shows a favorable behavior. Empirical investigations show that already for random sampling results similar to those for the favorable sample sets are obtained. Furthermore, hints are given for suitable sampling at ridges and sharp edges.

Part III

Additional Application Contexts of the Reconstruction Algorithm

Chapter 10

Interactive (Re-)construction

The reconstruction algorithm described in the preceding chapters is designed so that it should yield satisfactory reconstructions if the sample set fulfills certain conditions. However, in practice such conditions can not always be expected to hold, and in that case interactive intervention by the user may support the algorithm. Another aspect is that the reconstruction algorithm can also be used as modeling algorithm. In computer-aided geometric design, a widespread approach is to use control points from which a shape is interpolated or approximated. An interesting observation is that the reconstruction algorithm can be used for that purpose, too. In this chapter, we outline possibilities of those two aspects, and give further examples which illustrate the behavior of our algorithm.

10.1 Aspects of Interactivity

Interactive reconstruction means to observe and influence the behavior of a reconstruction algorithm interactively. There are at least two basic possibilities.

The first one is *interactive choice of the parameters* which control the behavior of the algorithm. The probably most simple case of interactivity is to choose a set of parameter values, start the algorithm, inspect the result, modify the parameters if the result is not satisfying, perform another run of the algorithm, and iterate this process until a reasonable result is achieved.

The second possibility is *interactive manipulation of the point set*, that is insertion or deletion of points. There are several interesting application scenarios of this type of interaction:

Correction in the case of not satisfying reconstruction: A reason for a not satisfying reconstruction may be unproper sample points, for example too few of them, in particular at sharp feature lines of a surface. This deficit may be remedied by insertion of further points at critical locations.

Interactive digitization: The reconstruction algorithm is executed during interactive acquisition of sample points, for example by tactile devices like a robot arm. The choice of sample points is adapted to the observed requirements of the algorithm.

Surface modeling: Surface modeling is a generalization of interactive digitization, in that no physical surface has to be present. The user just inputs 3D points which are on the virtual surface he is designing.

In the following we will focus on the interactive manipulation of the point set.

The following degrees of interactivity can be distinguished:

1. Interactive insertion or deletion of points, and reconstruction ”in batch” if the process of insertion is finished.

2. Interactive insertion or deletion of points, and online reconstruction and visualization.

From a computational point of view, the first possibility is easy because the algorithm can be used as is. It demands some experience of the user with the reconstruction algorithm in order to know how to set the points to avoid a high number of possibly time-consuming runs of the algorithm.

Online updating after every insertion or deletion in the second case can be performed according to two main strategies:

1. Recalculation from scratch.
2. Update of the current graph.

From the view of interactivity the first approach is acceptable only if the calculation time is short. Short calculation times are achieved for small point sets, but become difficult for large sets. Time can be saved by applying a hierarchical concept. First, just the clustered 1-environment graph (1-EG) is constructed. If the constructed surface description graph looks sufficiently well, finally the triangulation is included. That approach is reasonable and acceptable because a good reconstruction can only be achieved if the earlier graphs already are reasonable. In many cases triangulation from the surface description is canonical, and no or just little further manipulation of the point set is necessary in order to get a satisfying result. As we already have seen in Table 8.1 of the previous chapter, the computation of the clustered 1-environment graph needs just a few seconds if it is computed from scratch. However, the interactive update is a slightly different task which usually means that only a small part of the complete graph has to be recomputed. Therefore, even if the point sets get bigger the update procedure can still have real-time behavior.

Update of the current graph means to provide a data structure which efficiently supports the following operations:

Vertex insertion:

Given: A set of points and a reconstruction, for example an EG.

Wanted: For an arbitrary additional point, an update of the reconstruction which includes the new point.

Vertex deletion:

Given: A set of points and a reconstruction, for example an EG.

Wanted: For an arbitrary point of the set, an update of the reconstruction without the selected point.

A simplification is possible by avoiding explicit vertex deletion. The reason is that deletion may happen only rarely, because slight faults may also be corrected by insertion of further points. If necessary, vertex deletion can be implemented also on a data structure which just supports insertion [Ove83]. One approach is to store a sequence of intermediate states of the data structure, and the sequence of insertions between two consecutive of them. The deletion can be performed by going back to the state immediately before the insertion of the vertex to be deleted, and process a sequence of insertions from that point, without inserting the deleted point. This approach works efficiently, if vertices not too far in the past are eliminated.

More on dynamic data structures for surface description graphs, can be found in Section 10.4.

In the following we give two case studies of interactive modeling from scratch and interactive selective reconstruction on a given point set.

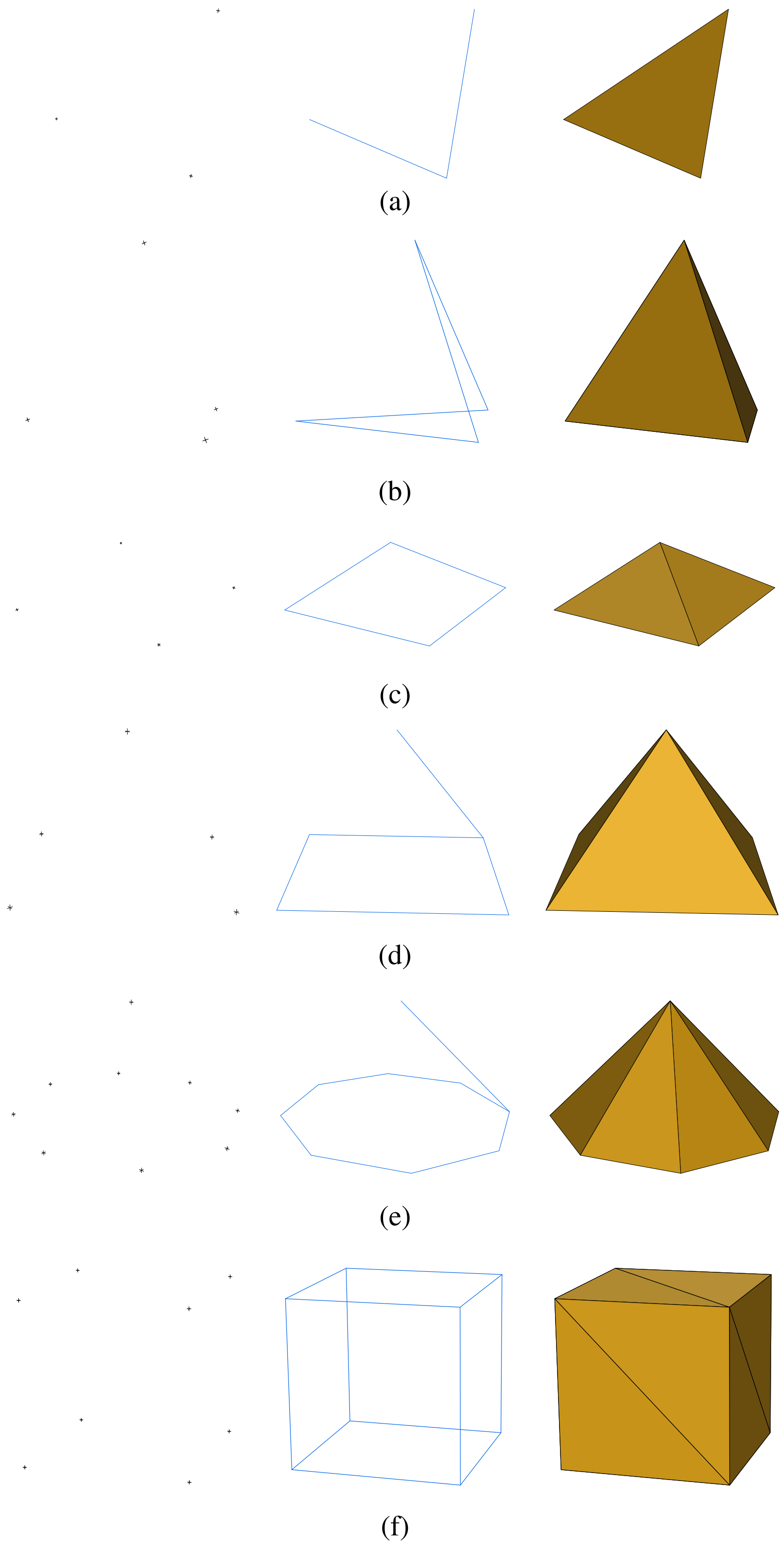


Figure 10.1: Construction of a triangle, a tetrahedron, a flat tetrahedron, a pyramid, a "polygonal" cone, and a cube.

10.2 Interactive Modeling from Scratch

Modeling from scratch means to start with an empty space and to build up a triangular manifold step by step by insertion or deletion of points, and application of the surface reconstruction algorithm on the resulting sets of points. In the following we illustrate the possibilities of that surface modeling approach at examples.

Figure 10.1 (a) shows the probably most simple example, the modeling of a triangle by three points. If the three points are arranged so that the resulting triangle has an angle larger than the boundary control parameter γ_c' of the algorithm then no triangle is returned.

The reconstruction of a tetrahedral surface is more complicated. The reason is that it depends on the parameter δ_c which bounds the dihedral angle, and like for the triangle, on the parameter γ_c'.

Figure 10.1 (b) shows a reconstruction of all faces of a tetrahedron.

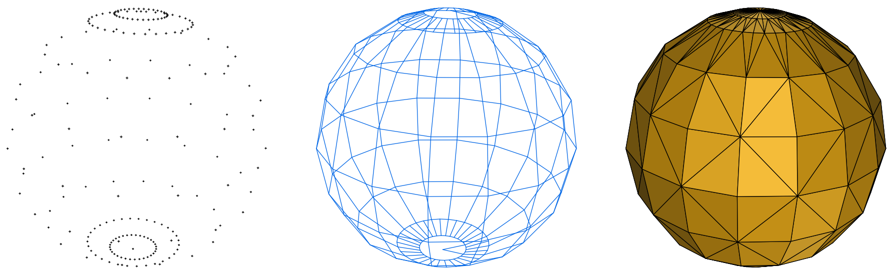


Figure 10.2: Construction of a sphere. The figure shows the given point set, the corresponding 1-environment graph, and the final result.

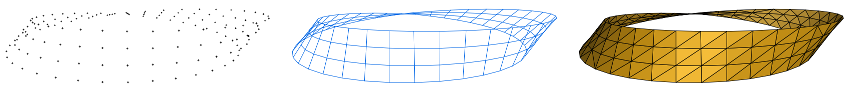


Figure 10.3: Construction of a Moebius strip: the point set, the clustered 1-environment graph, and the final Moebius strip.

In contrast to the previous figure, Figure 10.1 (c) depicts a reconstruction from four points which are almost in a common plane. Whether a closed surface without boundary is constructed depends on the parameters of the algorithm, like the dihedral angle bound δ_c. The canonical reconstruction is a surface with boundary, consisting of two adjacent triangles.

The examples show that it makes sense to offer also the parameters of the algorithm at the user interface for interactive manipulation. They also show that even with a very small number of points a desired shape can be achieved. In case of troubles, more points usually help to overcome the difficulties.

Figure 10.1 (d) shows the successful construction of a reasonably shaped four-sided pyramid, and Figure 10.1 (e) presents a "polygonal" cone of similar shape. Here troubles concerning the closeness of the surface may arise if the tip of the pyramid or the cone is close to the base.

Figure 10.1 (f) demonstrates that a cube can be constructed, too, from its eight vertices. Figure 10.2 shows the approximation of a sphere.

The application of the algorithm is not limited to orientable surfaces. Figure 10.3 shows a point set sampled from a Moebius strip and the corresponding reconstruction by the algorithm.

Figures 10.4 and 10.5. show that sharp edges can be modeled, too. The first figure illustrates that sharp surface edges that are randomly sampled can be reconstructed appropriately. The 1-environment graph

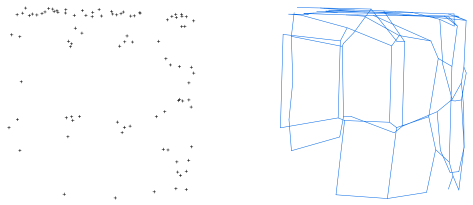


Figure 10.4: Construction of an edge of changing curvature: the point set, the clustered 1-environment graph, and the final result.

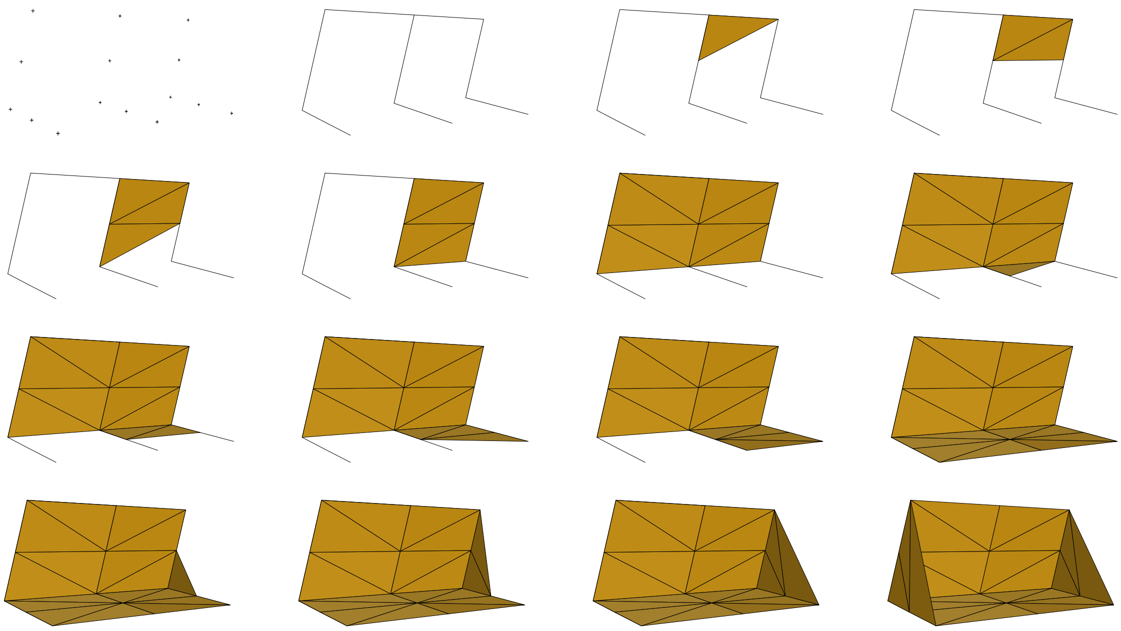


Figure 10.5: Construction of a sharp edge: the point set, the EMST taken as surface description graph, intermediate results of triangulation after 1,2,3,4,8,9,10,11,12,16,17,18, and 19 inserted faces, and the final reconstruction result (bottom right).

adapts naturally with edges to the situation so that the reconstruction algorithm can set surface triangles around the edge. For demonstration purposes, in the second figure the EMST is taken as surface description graph. Because the EMST contains less graph edges than the clustered 1-EG it is usually more difficult for the algorithm to treat the sharp surface turn of this example correctly. Therefore, the reconstruction of the artificially sharp surface turn of this example can be better analyzed. We see that no surface triangle is set in the interior of the object although the sector angle between the edges is very small. Obviously, the greedy triangulation with its sector selection strategy prefers the insertion of triangles that fit smoothly into the surrounding current mesh structure. This behavior has prevented the algorithm from inserting ”wrong” triangles into the surface.

As we know, the reconstruction algorithm works in two phases, construction of a surface description graph, and triangulation of this graph. In an interactive environment it may be reasonable to replace the environment graphs used by the algorithm by an explicitly edited arbitrary graph. The advantage may be that the user can draw the ”right” edges directly.

The reconstruction of Figure 10.6 shows that cylindrical structures can be constructed by generating circle-like structures by just a few points, and by arranging them so that they are connected by edges of the EMST or the environment graph. For our example, it is of course hard to say which kind of surface

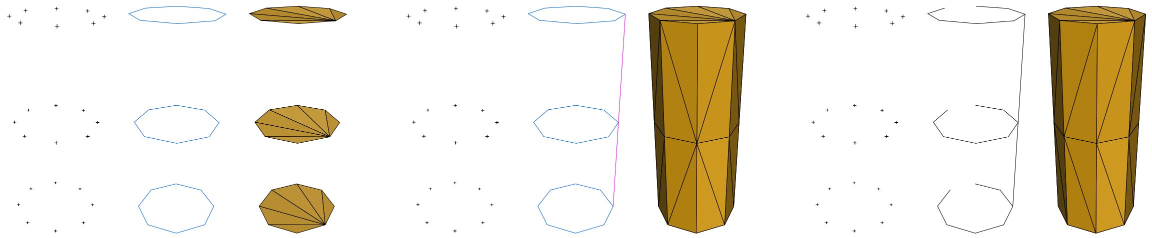


Figure 10.6: The reconstruction of cylindrical structures. Left: the point set with its clustered 1-environment graph and the final reconstruction without user interaction. Middle: the rings of graph edges have been connected by user interaction so that the cylindrical structure could be reconstructed. Right: the EMST has been taken as surface description graph which is also sufficient in order to reconstruct this type of object.

is represented by the point rings. If the clustered 1-environment graph is used as surface description graph, every point ring describes a single circular surface part, cf. Figure 10.6 (left). If the point rings are connected by the user with two edges (magenta) as shown in Figure 10.6 (middle), the structure for a complete cylinder is given, and the algorithm indeed constructs a complete cylinder.

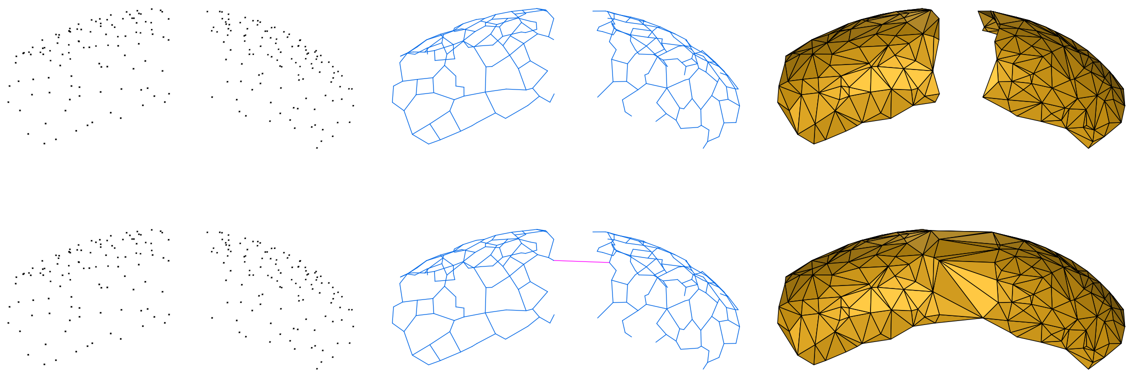


Figure 10.7: Top: Two parts of a surface have been sampled individually. The reconstruction yields two disjoint surface meshes. Bottom: Just one single edge (magenta) between points of the two parts must be inserted in order to combine both parts to a single surface.

Additionally, this cylindrical object is also a good example to show that even the EMST itself is a good and sufficient surface description graph. Because the EMST connects all points with graph edges it assures that the point rings are connected with each other so that the reconstruction algorithm can generate a connected surface, cf. Figure 10.6 (right). This example shows that in interactive environments the EMST and the clustered 1-environment graph might be applied by the user.

Another situation of simple user interaction is shown in Figures 10.7 and 10.8 where two surfaces are joined by inserting just a single but significant graph edge. The SDG displayed in the upper middle of Figures 10.7 and 10.8 does yield just edges in order to reconstruct the surfaces for the left and right part separately. The lower middle part of both Figures shows that by inserting only one edge the two surface parts can be joined.

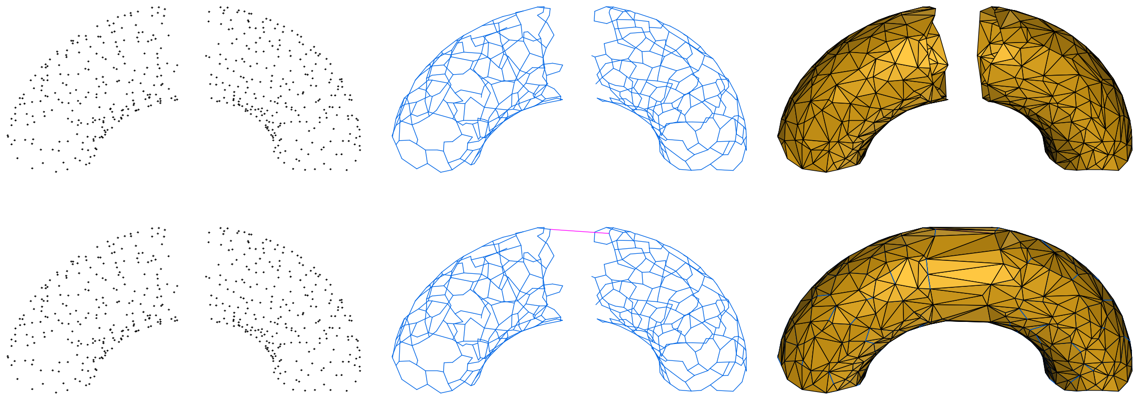


Figure 10.8: A similar situation as in Figure 10.7. Top: The two surfaces are tubes which have been generated independently by the reconstruction algorithm because they are not connected by graph edges. Bottom: The insertion of just one edge (magenta) yields enough information for the reconstruction algorithm in order to combine the two tubes to one single tube.

10.3 Interactive Selective Reconstruction

Selective reconstruction means the restriction of the application of the reconstruction algorithm to a subset of the set of sample points. Selective reconstruction is useful if only small parts of a large data set are of interest. An example is that just one part of the data set is useful while other parts are distorted by noise.

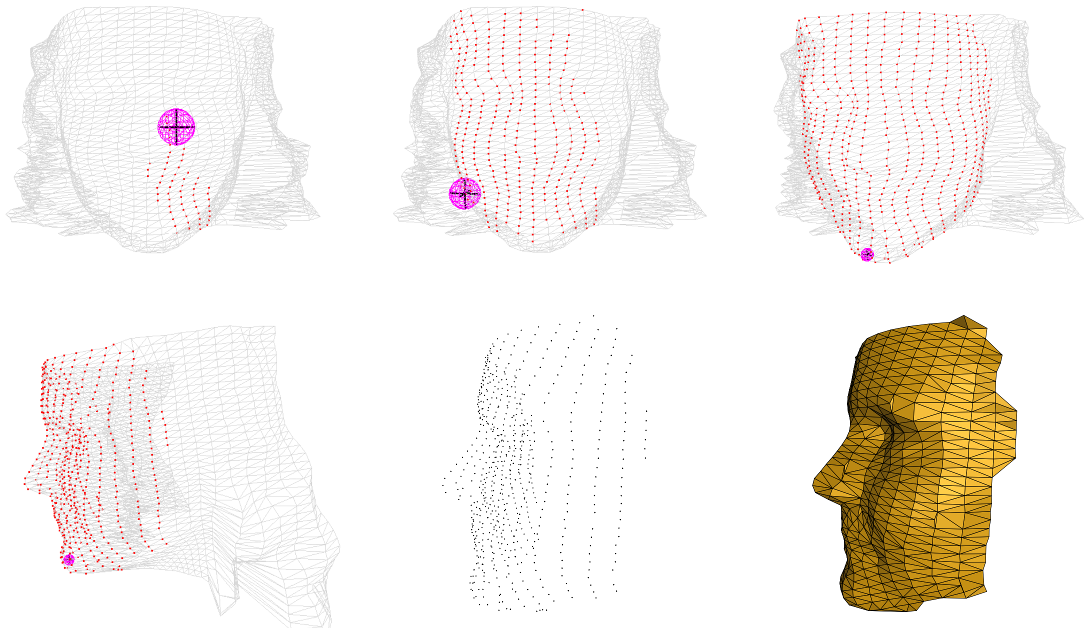


Figure 10.9: Explicit selection of a subset of points by a so-called "picking sphere". All points inside the picking sphere become part of the point set picked.

We distinguish between two versions of selective reconstruction which differ in the approach of selection of the subset:

1. Explicit selection of a subset of points which is subject of reconstruction by the surface reconstruction algorithm.

2. Design of a surface description graph by using points of the given point set, according to one of the versions of modeling from scratch. From the surface description graph, the manifold can be constructed by one of the following two possibilities:

   (a) Direct usage of the graph for triangulation.

   (b) Transformation of the graph into a feasible environment graph under consideration of the given point set.

An interesting aspect of the first alternative is the method of interactive point selection. Clicking points individually is tedious for larger point sets. As an alternative we use a *selection sphere*. The selection sphere is moved in space, and all those points are selected which have been in the interior of the sphere during the motion. The radius of the sphere can be changed interactively so that its sensitivity can be adapted dependent on the details of the region subject to current selection.

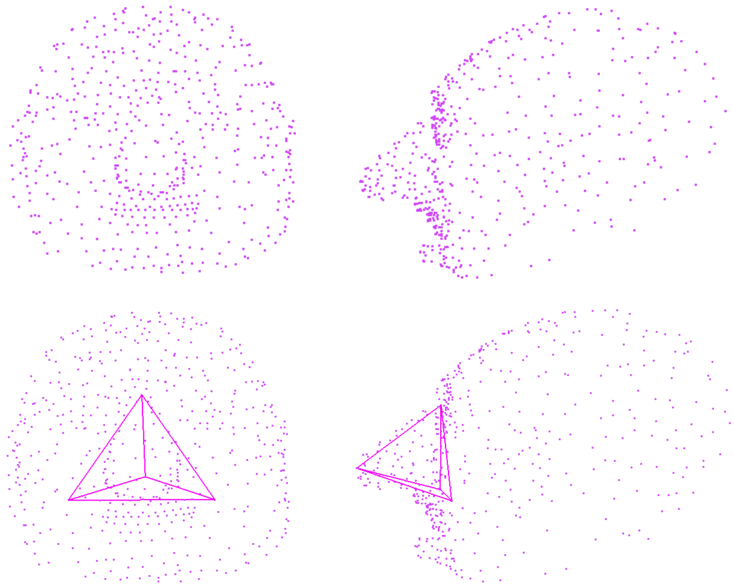


Figure 10.10: Top: The point set of the head of a puppet in front and side view. Bottom: The user decides to reconstruct the nose. The reconstruction region is determined by surrounding the area with six edges, arranged in the shape of a tetrahedron, which take into consideration the extreme convexity of the nose.

Figure 10.9 shows snapshots of a selection process and the result of reconstruction on the selected point set. This approach has shown quite useful in particular in combination with a stereoscopic display and a direct 3D-input by computer-vision-based hand tracking [Blu97, Koh99, SK95].

The background of version 2 (a) is to sketch just a rough wire frame over a subset of points, and the reconstruction algorithm fills the wire frame in its triangulation phase to a triangular manifold. The idea behind version 2 (b) is to automatically adapt the sketched graph to the other points of the point set which is a kind of "projection of the graph to the point set". This might allow rough sketching, possibly with long edges, which are then algorithmically refined into edges of an environment graph.

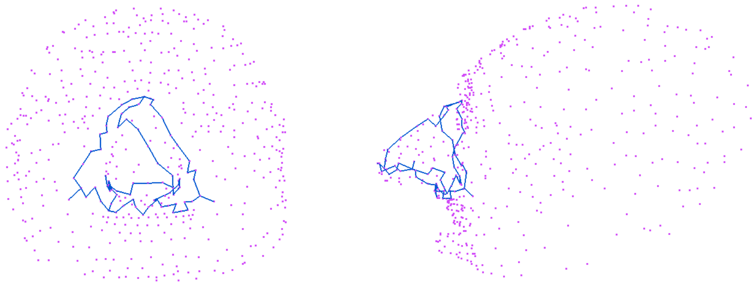


Figure 10.11: The set of edges after the refinement process. The new edges are scattered over the surface.

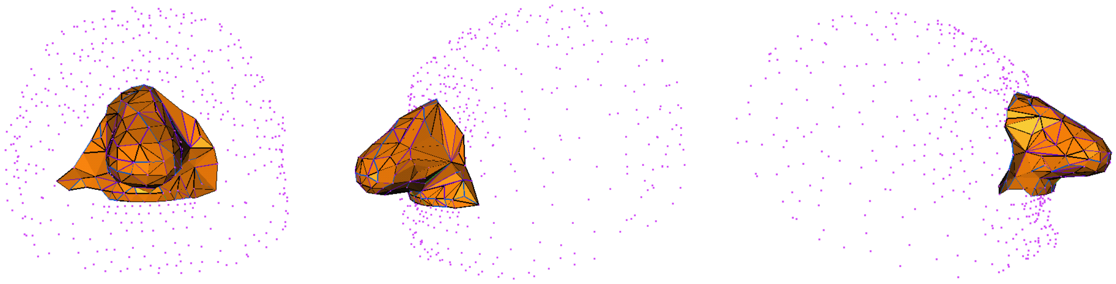


Figure 10.12: The result of reconstruction from the refined edges. The area where the reconstruction took place is a good approximation of the area that has been described by the user with the initial six edges, cf. Figure 10.10.

The algorithm of edge refinement works as follows. A set of candidate edges is maintained which initially consists of all edges of the sketched graph which do not have an empty β-environment. The algorithm iteratively takes edges $e = \overline{\mathbf{pq}}$ from that set. A vertex $\mathbf{r}$ in the β-environment of e with smallest sum of distances to the vertices $\mathbf{p}$ and $\mathbf{q}$ of e is chosen. Then e is replaced with the two new edges $\overline{\mathbf{pr}}$ and $\overline{\mathbf{qr}}$. Those of the new edges which do not have an empty β-environment are inserted into the set of candidate edges. The algorithm terminates when the set is empty. For $0 \leq \beta \leq 1$, the algorithm terminates because the new edges are shorter than the original ones.

Figures 10.10 and 10.11 show an application of the refinement algorithm to a larger point set. Figure 10.10 shows the sketched wire-frame consisting of six edges arranged in the form of a tetrahedron. Figure 10.11 depicts the transformation into an 1-environment graph by the edge refinement algorithm. The final result of reconstruction is displayed in Figure 10.12.

10.4 Computational Issues

Possibilities of calculation of environment graphs have been described in Section 5.3. In the following we discuss algorithmic possibilities of updating clustered environment graphs.

The principle idea for updating the clustered β-environment graph is to re-calculate it only in the region where it could change its structure because of a point insertion/deletion. For this purpose, the knowledge on its different calculation phases is taken into account.

Updating starts with an already existing clustered β-environment graph G for $\beta \leq 1$ of an initial point set P as defined algorithmically in Section 5.2.2. The first two phases of the algorithm deliver a radius for every vertex $\mathbf{p}$ in the end. Let $r_1(\mathbf{p})$ be the radius resulting from the first phase, and $r_2(\mathbf{p})$ the radius of the second phase.

Further, let $r_{1,\beta}(\mathbf{p}) > r_1(\mathbf{p})$ denote the first distance of a point $\mathbf{p}$ to another point $\mathbf{q}$ with $d(\mathbf{p}, \mathbf{q}) > r_1(\mathbf{p})$ for which the β-environment $E_\beta(\mathbf{p}, \mathbf{q})$ is not empty of points. If no such point $\mathbf{q}$ for a point $\mathbf{p}$ exists, then $r_{1,\beta}(\mathbf{p}) := \infty$.

10.4.1 Point Insertion

In this section the update procedure for a clustered β-environment graph is described if a new point $\mathbf{p}_{new} \notin P$ is added to the current point set P. Let $P' := P \cup \{\mathbf{p}_{new}\}$ denote the new point set and $r_1'(\mathbf{p}), r_2'(\mathbf{p})$ denote the new radii with respect to steps 1 and 2 of the original clustering algorithm for points $\mathbf{p} \in P'$. The *update process for point insertion* consists of three steps which are related to those of the original clustering algorithm. Let 1',2', and 3' denote these new steps which are described below:

1'. In the first step of the update procedure all radii $r_1(\mathbf{p})$ that could change because of insertion of $\mathbf{p}_{new}$ are updated to new radii $r_1'(\mathbf{p})$ as described here.

   Let H be the *initial heap* of points consisting of the following points:

   - the new point $\mathbf{p}_{new}$,
   - all points $\mathbf{p}$ with $\mathbf{p}_{new}$ in their $r_{1,\beta}(\mathbf{p})$ radius, that is, for which $d(\mathbf{p}, \mathbf{p}_{new}) \leq r_{1,\beta}(\mathbf{p})$.

   Then, the original step 1 of the clustering algorithm is applied to all elements of H so that each element $\mathbf{p}$ of H receives a new radius $r_1'(\mathbf{p})$.

   All other points $\mathbf{q} \in P' - H$ inherit their old radius of the non-updated graph, so that $r_1'(\mathbf{q}) := r_1(\mathbf{q})$.

   Then, H is extended by all points $\mathbf{p} \in P'$ whose spheres of the old radius $r_2(\mathbf{p})$ contain $\mathbf{p}_{new}$, that is, for which $d(\mathbf{p}, \mathbf{p}_{new}) \leq r_2(\mathbf{p})$.

   As initialization for step 2' all radii $r_2'(\mathbf{p})$ for $\mathbf{p} \in P'$ are initialized to $r_2'(\mathbf{p}) := r_1'(\mathbf{p})$.

2'. Now, the second part of the update procedure is applied which consists of iterative application of this step 2' onto a changing heap H until a certain final state is achieved.

   The original step 2 of the clustering algorithm is applied under consideration of the radii $r_2'(\mathbf{p})$ for all points $\mathbf{p} \in P'$ and with restriction to the elements of H, that is, for each considered point pair at least one point must be part of H. This results in the current new radii $r_2'(\mathbf{p})$ for each point $\mathbf{p} \in P'$.

   The set of elements $\mathbf{q} \in P' - H$ which have an intersection with a point of $\mathbf{p} \in H$ in the way $d(\mathbf{p}, \mathbf{q}) \leq \max(r_2(\mathbf{p}), r_2'(\mathbf{p})) + \max(r_2(\mathbf{q}), r_2'(\mathbf{q}))$ is called $\hat{H}$. The maximum of the radii r_2, r_2' for $\mathbf{p}$ and $\mathbf{q}$ is taken in order to consider definitely all points that could induce either a radius update or a new virtual edge.

   After that, the original step 2 of the clustering algorithm is applied to the elements $H \cup \hat{H}$ with the same restrictions as before. This ensures that all elements of $\hat{H}$ receive a radius update not only from points of H but also from their other surrounding points of P'.

   If then for all points $\mathbf{p}$ of the current set $\hat{H}$ the new radius $r_2'(\mathbf{p})$ is equal to the old radius $r_2(\mathbf{p})$, that is $r_2(\mathbf{p}) = r_2'(\mathbf{p})$ for all $\mathbf{p} \in \hat{H}$, then the update process for step 2' is complete. Otherwise, step 2' is repeated again after the heap H has been extended by the following points:

   - all points $\mathbf{q} \in P' - H$ which are contained in the $r_2(\mathbf{p})$-radius of elements $\mathbf{p} \in H$ and whose $r_1(\mathbf{q})$-radius is smaller than $r_2(\mathbf{p}) - d(\mathbf{p}, \mathbf{q})$,

- all points $\mathbf{q} \in P' - H$ which fulfill the relation $d(\mathbf{p}, \mathbf{q}) \leq \max(r_2(\mathbf{p}), r'_2(\mathbf{p})) + \max(r_2(\mathbf{q}), r'_2(\mathbf{q}))$ with a point $\mathbf{p} \in H$ and for which $r'_2(\mathbf{q}) \neq r_2(\mathbf{q})$.

3'. After the iteration process of step 2' has taken place, all edges incident to points of H are deleted. Then, the virtual edges that have been generated in step 2' are inserted in order of increasing length using the χ-intersection test. If a new edge χ-intersects an old edge $\overline{\mathbf{pq}}$ that connects two points $\mathbf{p}, \mathbf{q} \notin H$, then $\mathbf{p}, \mathbf{q}$ are put onto H, all new already inserted edges are removed, and the iteration process of step 2' of the *update procedure for point insertion* is re-started. This case is very unlikely, since in that case $\mathbf{p}, \mathbf{q}$ usually would have been close to points of H and therefore with probability also part of H. If only χ-intersections occur with new inserted edges of this step 3' then the update process of the clustered β-environment graph is complete after all virtual edges have been considered.

10.4.2 Point Deletion

All points $\mathbf{p} \in P$ with the property $d(\mathbf{p}, \mathbf{p}_{del}) \leq r_2(\mathbf{p}) + r_2(\mathbf{p}_{del})$ with respect to the point $\mathbf{p}_{del} \in P$ to be deleted are put onto an initial heap H.

Then, the new point set P' is defined by $P' := P - \{\mathbf{p}_{del}\}$ and all edges of the current graph that were incident to $\mathbf{p}_{del}$ are deleted.

After that, the same update procedure as for point insertion is applied with respect to the different initial heap H.

10.4.3 Tetrahedrizations for Speed-up

As in Section 5.3, tetrahedrizations are useful for speeding-up the calculation. The following tasks have to be performed.

The tetrahedrization has to be updated with respect to the manipulated point, that is the point has to be inserted or deleted. For the Delaunay triangulation, in practice common libraries like the CGAL [CGA] can be used for that purpose. Unfortunately, up to now there does not seem to exist any library which handles dynamic deletion of single points from a Delaunay tetrahedrization. For the hierarchical tetrahedrization insertion does affect just one tetrahedron. Deletion is problematic, too. A possibility to treat the problem of point deletion is to leave deleted points in the tetrahedrization and to label them as de-activated. This approach, however, does not work for the supergraph approach based on Delaunay triangulations mentioned in Section 5.3.

Each time a point is inserted or deleted all points that have "visited" the modified tetrahedra during their previous k-nearest neighbor computations have to be re-initialized for this task (Appendix B).

During point insertion, the points with mutually intersecting spheres are identified by processing the list of points having visited a tetrahedron during a k-nearest-neighbor query, and by comparing their associated radii. Point deletion is performed analogously. All those points are considered for intersecting spheres which have visited the deleted point $\mathbf{p}$ in previous nearest neighbor queries.

10.5 Discussion

The surface-oriented approach of our reconstruction algorithm makes it to a useful tool for interactive surface modeling. The surface description graphs provide an efficient computable impression of the final shape to be expected, and thus are quite useful as control structure of the shape. As surface description graph, the suggested "automatically" calculated environment graphs, but also graphs interactively designed or modified by the user can be applied.

Another benefit is that a locally-defined reconstruction can be achieved by ”projecting” graphs drawn by the user onto the point set. This approach reduces the interaction time of the user for selection of the desired subregion subject to reconstruction.

Chapter 11

Noise Elimination

A sample set is called *noisy* if the sample points deviate from the surface by a small, usually random, distance. Noise in sample sets may be caused by technical properties of the sampling device. If the sample points deviate considerably from the surface, it might happen that a reconstruction algorithm that is relying on criteria like the dihedral angle or on neighborhood criteria which assume the surface to be sufficiently smooth might not longer yield reasonable results.

A powerful method of noise reduction is Laplacian smoothing of second order. This method is originally defined for distorted vertices of a triangular mesh. In our application, however, we do not have a mesh before reconstruction. We solve this problem by defining an auxiliary mesh on the given sample set to which the smoothing operator can be applied for noise reduction.

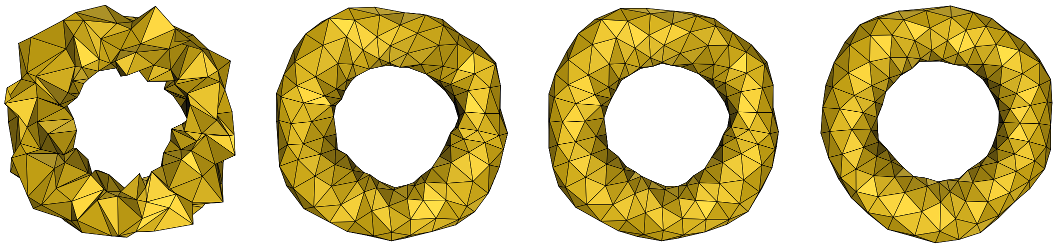


Figure 11.1: From left to right: The original noisy surface mesh, and the de-noised mesh after 1, 2, and 3 iterations of Laplacian smoothing of second order. Here, the values $\alpha = 0$ and $\beta = 0.5$ have been chosen. Note, that even after just one iteration the de-noised mesh has significantly improved.

11.1 Laplacian Smoothing of Second Order

For meshes, the probably up-to-now best smoothing filter is Laplacian smoothing of second order [Tau95, KCVS98, VMM99, Vol98]. *Laplacian smoothing of first order* considers the set $\mathrm{adj}(\mathbf{q})$ of adjacent vertices of every vertex $\mathbf{q}$ of the mesh, and moves it to a new location $\mathbf{p}$ which is calculated by averaging the vertices in $\mathrm{adj}(\mathbf{q})$,

$$\mathbf{p} := \frac{1}{n} \sum_{\mathbf{q}' \in \mathrm{adj}(\mathbf{q})} \mathbf{q}'.$$

For *Laplacian smoothing of second order*, basically the same procedure is applied to the difference $\mathbf{d}$ between the new and the original vertex location. This is performed with the slight modification that at iterative application both the location $\mathbf{q}$ at the current level of iteration and the location $\mathbf{o}$ of the vertex in the original mesh are considered by an α-weighted average, as it is done for the differences with a weight β, too:

$$\mathbf{d} := -\left(\beta\mathbf{b}(\mathbf{q}) + \frac{1-\beta}{|\mathrm{adj}(\mathbf{q})|}\sum_{\mathbf{q}'\in\mathrm{adj}(\mathbf{q})}\mathbf{b}(\mathbf{q}')\right)$$

where

$$\mathbf{b}(\mathbf{q}) := \mathbf{p} - (\alpha\mathbf{o} + (1-\alpha)\mathbf{q}).$$

The result of Laplacian smoothing of second order of a vertex **q** is the sum of **p** and **d**.

The purpose of distance correction by adding **d** is to reduce the shrinking effect of Laplacian smoothing of first order.

This formula is applied iteratively until a satisfying result is achieved.

The value α weights the influence of the original noisy points. While this is sometimes desired when existing surface meshes have to be smoothed, the influence of the original points during noise elimination for arbitrary point sets is not favorable. Since the original point positions might have been generated because of heavy noise the restriction to these positions does not make sense. Therefore, for this application $\alpha = 0$ should be chosen. For the other parameter the value $\beta = 0.5$ is a good choice [VMM99].

Figure 11.1 shows an example of an application of Laplacian smoothing of second order to an artificially noised data set, for $\alpha = 0$ and $\beta = 0.5$.

For mesh smoothing it is of advantage to alternate the application of a surface smoothing operator like Laplacian smoothing of second order with smoothing by edge swapping [FOG97, DLR90, BS91, Bro91, Ham97].

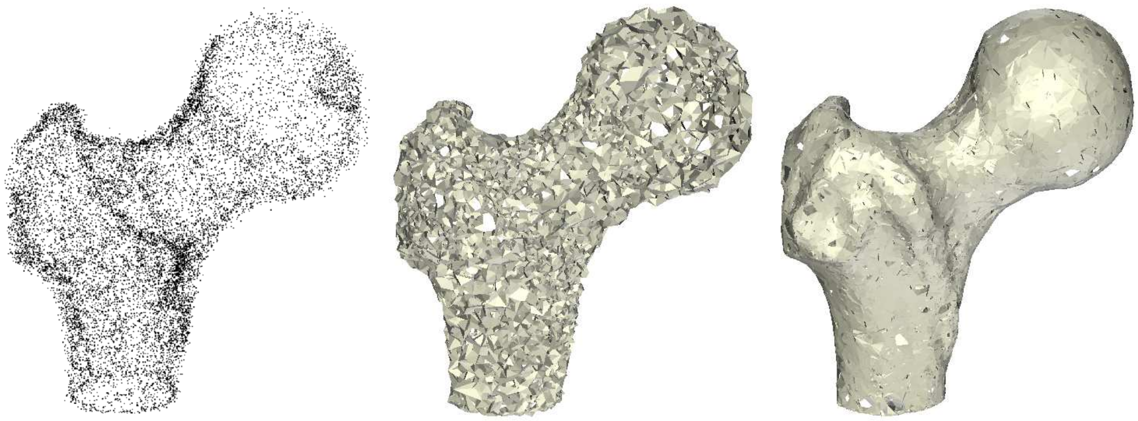


Figure 11.2: From left to right: a noisy point set, the auxiliary mesh constructed from it, and the auxiliary mesh smoothed by alternating the application of Laplacian smoothing of second order and edge swapping.

11.2 Direct Noise Elimination by Auxiliary Meshes

The difficulty in our setting is that no mesh is present with the given sample points. The idea to cope with that problem is to construct an auxiliary mesh. The auxiliary mesh needs not to be a perfect reconstruction of the surface, but it should reflect the neighborhood relations between the given sample points approximately. The basic approach is as follows [VMM99]:

1. Calculate a surface description graph (SDG) using the first phase of the reconstruction algorithm of the previous chapters.

2. If the vertex degrees of the resulting graph are not typical for a surface mesh, augment the set of adjacent vertices of every vertex by introducing edges up to a suitable number of vertices in its neighborhood not yet adjacent. In order to take care of variations of the point density, an adaptive number of nearest neighbors is used, which is estimated by using the length of incident SDG edges as reference value for the different vertices.

3. Around each vertex a corona of faces is created on base of incident edges like a spanning umbrella. In this manner for each face one new edge is created, if it does not already exist. These edges are important for the connectivity of the new mesh.

4. Faces not fulfilling a given mesh quality measure are removed. A quality measure introduced by Bank and Smith [BS97] is used for that purpose.

5. Remove triangles so that at most two triangles are incident to every edge in order to make edge swapping applicable, under preservation of pairs of triangles with large dihedral angles. All edges which do not have at least one incident face are removed.

Step 5 is necessary because the set of triangles generated in the previous steps may be non-manifold.

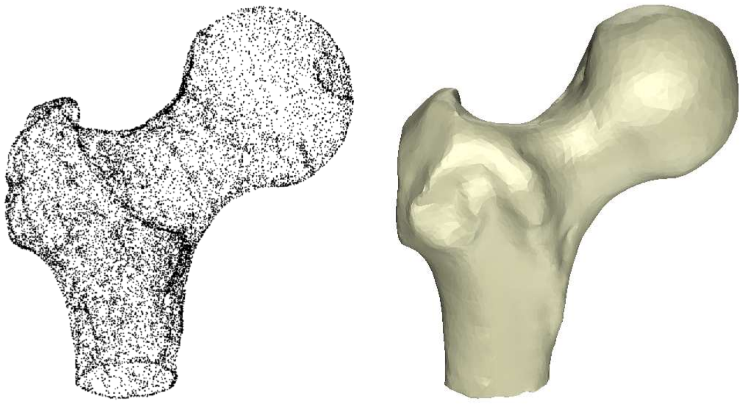


Figure 11.3: Reconstruction based on the noise-reduced set of points shown at the left.

Figure 11.2 shows a distorted sampling set of a bone, the derived auxiliary mesh, and the result of smoothing of that mesh by an alternating application of Laplacian smoothing of second order and edge swapping. Figure 11.3 depicts the noise-reduced point set and the reconstruction of the bone obtained by our reconstruction algorithm from this point set.

11.3 Discussion

The approach of using an auxiliary mesh for smoothing of noisy point sets sampled from a smooth surface has turned out to be useful in practical examples. A topic of further research might be to extend the theoretical investigations of this thesis which assume the sampling points being on the surface, to sampling points only close to the surface. A triangular mesh with those noisy sampling points as vertices might be considered as reconstruction of the given surface, if it maps one-to-one under the NN-image to a non-overlapping mesh on the surface.

Part IV

Future Developments and Conclusion

Chapter 12

Future Developments and Conclusion

We have presented a surface (re-)construction algorithm which falls into the category of surface-oriented approaches. This category has found less interest in the past than volume-oriented approaches. A reason is that volume-oriented approaches usually can deliver well-defined surfaces that are topologically equivalent to the surface of a polyhedron. Surface-oriented algorithms have to invest some care on this issue, but immediately allow surfaces with boundaries.

Our algorithm consists of two steps. The first step is the construction of a surface description graph as a spanning skeleton. The task of the second step is the derivation of a surface by introducing chords into the surface description graph which define triangles.

A great advantage of this two-step-approach is that the basic shape can be controlled by the surface description graph, which is a new concept for surface reconstruction. The surface description graph can be constructed manually, automatically, or semi-automatically.

Interactive editing of the graph simplifies interactive input by the user in that less input is necessary than if a complete mesh has to be edited. The power of interactive editing of surface description graphs for the purpose of modeling surfaces has been demonstrated. A particular useful property in contrast to other approaches is that the surface reconstruction can be easily restricted to only subsets of the point set by very few modifications of the surface description graph.

We have found a suitable graph concept for automatic construction of surface description graphs: the β-environment graphs. These graphs are rather in-sensitive to the distribution of the given sampling points. They even work if the point set is not very dense.

The β-environment graphs can be forced to consider sharp edges and ridges by increased sampling density at those items. Such sequences of dense points form a special pattern to which the β-EGs are sensitive. A general question is whether there are other patterns and features in sampling sets which can be used to control the setting of edges in the surface description graph, or the choice of triangles in the subsequent triangulation, thus possibly achieving a more general graph concept.

The theoretical analysis of the algorithm is focused on the investigation of assertions for a favorable behavior of the reconstruction approach. It is ”heuristic” in that it gives mathematical arguments that the algorithm should behave as stated. One reason for this approach is that a rigorous mathematical treatment would be too extensive, even if it would be based on precise stochastic methods. For the requirements of practical application this is not really necessary. A stochastic treatment would require to model probability distributions. Usually treatable distributions do not model the real situation and thus the relevance of a precise treatment is of limited value. Our argumentation is that unfavorable configurations for the mathematical analysis occur rarely and even if they should appear in practice they do not affect the quality of the reconstruction. This means that the set of those configurations usually has less than full dimension in the space of all configurations, or has full dimension but is of small volume. This means that under the assumption of uniform distribution unfavorable configurations occur rarely. A closer look at the arguments, however, shows, that sampling sets can usually be chosen which are likely to avoid unfavorable situations. For example, a well-suited sampling strategy is to choose the sampling points as vertices of a quadrilateral mesh whose faces are moderately distorted squares. Additionally, our empirical investigations have shown that the unfavorable configurations are only unfavorable for the mathematical analysis and not for the reconstruction in practice. In all of our examples, the accuracy of the reconstruction in those cases is not influenced. Obviously, the triangulation criteria always choose surface triangles with a good approximation quality. This leads to the assumption that the triangles "inherit" their surface approximation quality from the surrounding triangles and it can be expected that mathematical arguments for this property can be found.

The analysis has been performed only for smooth surfaces, represented by so-called SF-surfaces S. The mathematical treatment of sharp edges is a problem for further research. An approach for a solution could be the consideration of singular surface points and what kind of sampling in those areas is needed and then extend this principle to surface ridges.

We have restricted our discussion to the existence of suitable sampling sets, not having the goal of sampling sets of minimum cardinality. A step in this direction would be to take into consideration the maximum possible SF-radius at every point, not just a constant radius r for the whole surface which bounds the occurring SF-radii from below. Our investigations can be generalized in the direction by replacing the constant r with a function $r : S \to \mathbb{R}_+$ for which $r(\mathbf{p})$ is at most equal to the maximum SF-radius at $\mathbf{p}$.

A further idea is to get independent from the quality of the sampling by using feature recognition in the surface description graph [MM98a, Mai98] if the graph itself does not contain enough surface information. The feature knowledge could be used in order to modify the graph so that the triangulation algorithm can generate a correct surface mesh even if the sampling was not sufficient. In order to improve flexibility in this approach, the recognition capabilities can be formulated in rules that can be also changed during run-time and interpreted by a rule evaluation system [Hei98].

As these remarks show, much has been achieved, but some interesting questions remain for future research. In conclusion, however, we believe that the approach presented in this thesis is a particular flexible and precise alternative to other approaches. It is well-suited for practical applications, even on non-dense data sets, and it allows useful assertions on the requirements of sampling.

Part V

Appendices

Appendix A

Implementation

The reconstruction algorithm has been implemented in C++ using the toolkits Tcl/Tk [Ous94], Tix [Lam93] and OpenInventor [Wer94]. These toolkits were taken because of their effectiveness in the design of graphical user interfaces as well as in the development of complex geometry viewers.

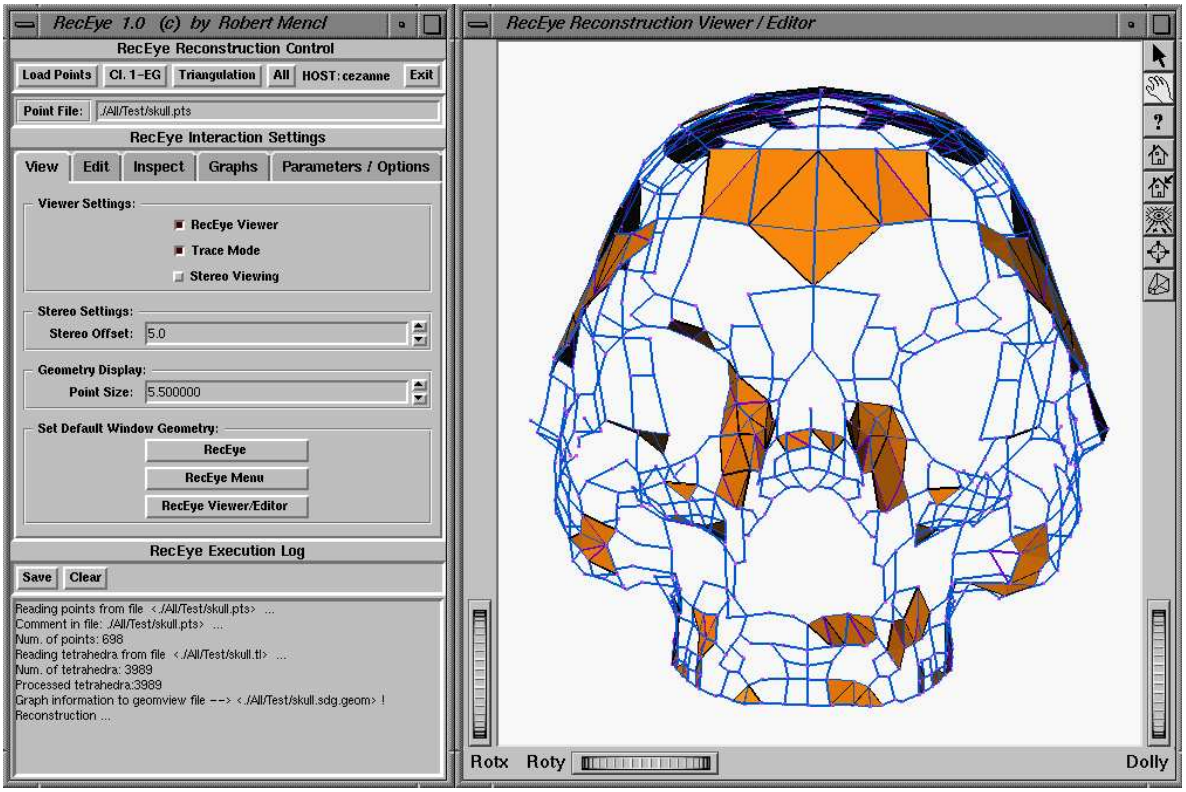


Figure A.1: A screenshot of the reconstruction system RECEYE. Left: the control panel. Right: the geometry viewer with an intermediate reconstruction.

The graphical user interface of the reconstruction system RECEYE is separated into a control panel and a geometry viewer window, cf. Figure A.1.

The control panel contains three sub-windows, the *Reconstruction Control*, *Interaction Settings*, and the *Execution Log* window.

The *Reconstruction Control* contains the basic buttons for the standard usage of the system. With *Load Points* the point set specified by the current file name <pointfile>.pts is loaded. The program expects an already existing tetrahedron file `<pointfile>.tl` which contains the Delaunay tetrahedrization of the point file. The first line of the file will be skipped, all other subsequent lines contain the point indices of the tetrahedra of the Delaunay tetrahedrization. For the Delaunay tetrahedrization the program DETRI of Mücke [Müc93b, Müc93a] can be used. The other buttons *Cl. 1-EG* (Clustered 1-EG) and *Triangulation* belong to the specific phases of the reconstruction algorithm. The (intermediate) results are saved using the geomview file format into the files `<pointfile>.sdg.geom`, for the graph, and `<pointfile>.s.geom` for the triangular mesh. The *Point File* and *Exit* buttons are self-explaining.

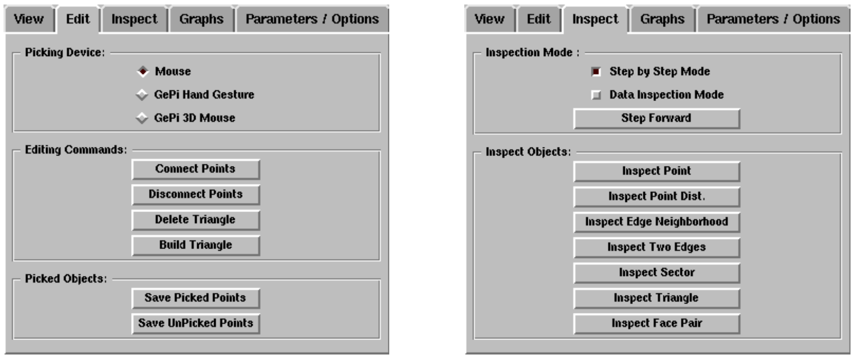


Figure A.2: The sub-menues for the interaction settings menues *Edit* and *Inspect*.

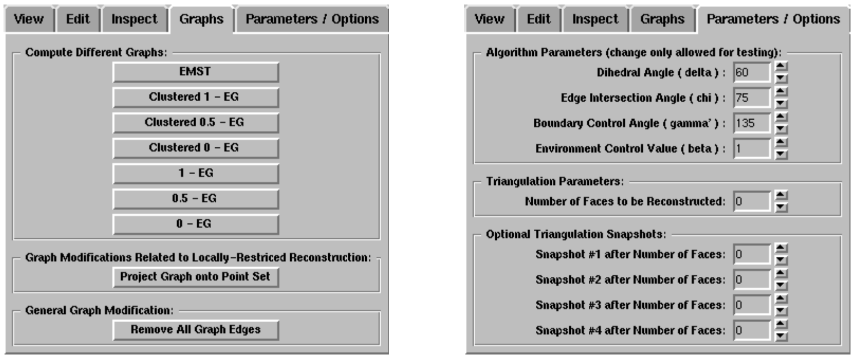


Figure A.3: The sub-menues for the interaction settings menues *Graphs* and *Parameters/Options*.

In the *Interaction Settings* the relevant data for the user interaction can be set. The menues are subdivided into 5 different categories: *View*, *Edit*, *Inspect*, *Graphs*, and *Parameters/Options*. An overview of these menues can be found in Figures A.1, A.2, and A.3.

In the following we give a short description of these five menues.

- *View:*
  - *Viewer Settings:* Here, the *Reconstruction Viewer* window, the *Trace Mode* for the reconstruction and stereo viewing can be switched on/off.
  - *Stereo Settings:* The eye distance offset of the stereo mode can be changed here.
  - *Geometry Display:* The size of the points in order to simplify picking can be increased.

  - *Set Default Window Geometry:* The default window geometry for both windows (*RecEye*), the control window alone (*RecEye Menu*), or just the reconstruction viewer window (*RecEye Viewer/Editor*) can be set.

- *Edit:*
  - *Picking Device:* The user can switch between the standard *Mouse*, the hand gesture picking (*GePi Hand Gesture*), and 3D movement of a picking sphere with simple mouse interaction as it is supported by OpenInventor [Wer94]. Picking with the simple mouse is performed in the picking mode of OpenInventor by first clicking onto the "arrow" in the upper right of reconstruction viewer window and then clicking on the points. In order to being able to turn the object in the reconstruction viewer window, the "hand symbol" in the upper right window has to be clicked. The hand gesture picking also moves a picking sphere through the point set. The set of points that is inside the picking sphere is picked. This hand gesture picking approach uses the library GEPI [Blu97] based on the hand gesture recognition system ZYKLOP [Koh99, SK95]. The description of the control of this system can be found in [Blu97].
  - *Editing Commands:* Pairs of points can be inserted as an edge (*Connect Points*) or their connecting edge can be deleted (*Disconnect Points*). Point triplets induce the deletion (*Delete Triangle*) and insertion (*Build Triangle*) of triangles. For the connection/disconnection of points and for the triangle deletion/insertion exactly two or three points, respectively, have to be picked.
  - *Picked Objects:* The picked points as well as their complement can be saved into the file `<pointfile>.[un]picked`.

- *Inspect:*
  - *Inspection Mode:* The *Step-by-Step* mode of the triangulation can be activated and each time a *Step Forward* by inserting one triangle after another, the reconstruction process can be investigated. In the *Data Inspection Mode* some additional information is printed which describes the internal computations for various modules of the system.
  - *Inspect Objects:* In this section the various objects which became important during the development of the reconstruction algorithm can be investigated.

- *Graphs:*
  - *Compute Different Graphs:* The EMST, the clustered β-EGs with $\beta = 1$, $\beta = 0.5$, $\beta = 0$ and their corresponding non-clustered β-EGs can be computed. All computed edges are added to the current graph.
  - *Graph Modifications Related to Locally-Restricted Reconstruction:* The "graph projection approach" of Chapter 10 for locally-restricted reconstruction can be applied here.
  - *General Graph Modification:* With the button *Remove All Graph Edges*, the current graph can be reset to the empty graph.

- *Parameters/Options:*
  - *Algorithm Parameters:* The parameters of the triangulation algorithm can be changed here.
  - *Reconstruction Parameters:* The number of faces to be inserted in one triangulation step can be modified here. If the value is 0 then the complete mesh is generated.
  - *Optional Triangulation Snapshots:* Four different values for the number of faces, where the current triangulation mesh has to be saved into a snapshot file `<pointfile>.shot#.s.geom` in the geomview file format.

The *Execution Log* window is used to provide the user with the relevant information for interaction. The button *Save* allows to store the contained information into the file `<pointfile>.receye.execlog`, and *Clear* deletes the displayed information on the screen.

Appendix B

Efficient and Flexible Nearest Neighbor Queries

Because of their occurrence in many applications, the efficient solution of nearest-neighbor problems has found particular interest in computational geometry in the past. Two main streams of approaches may be distinguished. One of them is centered around the concept of Voronoi-diagrams [PS85] the other one on more arbitrary spatial decompositions [AM91]. For solutions of the first stream, efficient worst case time bounds can often be proved, whereas for second type of approaches, the worst case behavior often is bad but they behave quite well in practice. Often, the heuristics are easier to implement than the more sophisticated Voronoi-diagram-based approaches. The solution of the k-nearest-neighbor problem falls in the second category.

In previous work on heuristic solutions of nearest neighbor problems [CK92, AMN$^+$94], regular space subdivisions play an important role. These subdivisions perform well on uniformly distributed data, but are somewhat less suited for data sets of strongly varying density. Triangulations, on the other hand, are an irregular data structure which adapts easily to all kinds of data distributions.

B.1 k-Nearest-Neighbor Search

The aim of the following is, as a result of the above considerations, to present an algorithm for solving the k-nearest-neighbors problem by taking advantage of a previously computed triangulation. A d-dimensional triangulation is defined as follows.

Definition B.1 (d-dimensional triangulation) *Let* $\mathbf{A}$ *be a real-affine space of dimension d with metric $d(\cdot,\cdot)$. An m-**simplex*** $\mathbf{s}$ *is the convex hull of $m+1$ points, called vertices, which are not contained in a $(m-1)$-dimensional subspace. A **subsimplex** of* $\mathbf{s}$ *is the convex hull of a proper subset of the vertices of* $\mathbf{s}$. *A $(d-1)$-subsimplex of a d-simplex is called a **facet**. Two d-simplices are called **incident** if one of them is a subsimplex of the other. Two d-simplices are called **adjacent** if they have d vertices in common and their intersection is a facet.*

We define the distance of a simplex $\mathbf{s}$ *from a point* $\mathbf{p}$ *as*

$$d(\mathbf{s},\mathbf{p}) := \min_{\mathbf{q}\in\mathbf{s}} d(\mathbf{q},\mathbf{p}) \quad .$$

Let the point set $P = \{\mathbf{p}_1,\ldots,\mathbf{p}_n\} \subset \mathbf{A}$ not be contained in a proper subspace of $\mathbf{A}$. *A **triangulation** T of P is a tesselation of the convex hull of P into d-simplices whose vertices are in P. We denote the subsimplices of all d-simplices of T collectively as the subsimplices of T. In particular, the points in P are the 0-subsimplices of T.*

The basic version of the problem we are treating is defined as follows.

Definition B.2 (k-nearest-neighbors query)

Input: *A vertex $\mathbf{p}_i$ of a d-dimensional simplicial decomposition T.*

Output: *The k-nearest vertex to $\mathbf{p}_i$ in T.*

Our algorithm for that problem can process multivariate data, as it functions in spaces of arbitrary finite dimension. Furthermore, it does not assume a particular metric. Many triangulation-based algorithms will work only with triangulations that possess certain properties, such as Delaunay triangulations. Our algorithm makes no such requisites. Thus, it can operate on any triangulation that another module has 'left behind.' In determining the k nearest neighbors, our algorithm explores only a part of the triangulation. While this part contains more vertices than just the k nearest ones, it is in general considerably smaller than the complete triangulation. The neighbor points are reported in order of increasing distance from the query point. In some applications, this order presents useful additional information. Implementation of the algorithm is straightforward. Apart from distance computations and the triangulation, only standard operations and data structures are needed.

With only minor modifications, the algorithm becomes applicable to slightly different types of queries. Thus, it is easy to find the data points lying within a certain radius from the query point. Not only vertices of the triangulation, but arbitrary points in the respective space can be used as query points. A query can be suspended after a certain number of neighbors have been determined, to be resumed later if further neighbors are needed. This is particularly useful for interactive graphical techniques where additional demand for neighborhood information arises as a result of feedback from the user.

The basic concept of our algorithm [WM96] is a ball which is centered at the query point $\mathbf{p}_i$ and whose radius increases continuously. As the ball expands, it encounters the vertices of T in order of increasing distance from $\mathbf{p}_i$. Our algorithm registers not only the vertices, but also the d-simplices of T in the order in which the ball encounters them. To this end, an appropriate subset of the d-simplices and vertices is stored in a heap, which is sorted by distance from the query point. The element closest to $\mathbf{p}_i$ is found at the top of the heap.

The expanding ball will, in general, encounter several d-simplices and/or vertices simultaneously. The algorithm, on the other hand, processes these elements one after another. At any given time during the expansion process, we call a d-simplex or vertex of T *intersecting* if the algorithm has determined that it intersects the ball. All other d-simplices and vertices are called *non-intersecting*, even if they do intersect the ball. The term closest is used with respect to distance from the query point.

Lemma B.3 *Let $\mathbf{p}_i$ be contained in at least one intersecting d-simplex. Then one of the closest non-intersecting d-simplices is adjacent to (i.e., shares a facet with) an intersecting d-simplex.*

Proof: Let $\mathbf{t}$ be a closest non-intersecting d-simplex, and let $\mathbf{q}$ be the point of $\mathbf{t}$ closest to $\mathbf{p}_i$. Since T covers a convex volume, it must cover the line segment $\overline{\mathbf{p}_i\mathbf{q}}$. By choice of $\mathbf{t}$ and $\mathbf{q}$, it is clear that each interior point of $\overline{\mathbf{p}_i\mathbf{q}}$ is contained in some intersecting d-simplex. Since we consider closed d-simplices, $\mathbf{q}$ is also contained in an intersecting d-simplex, say, $\mathbf{t}'$. (If the line segment has no interior points, then $\mathbf{q} = \mathbf{p}_i$ is contained in an intersecting d-simplex by hypothesis.) Now consider two interior points, $\mathbf{p}$ and $\mathbf{p}'$, of $\mathbf{t}$ and $\mathbf{t}'$, respectively. We choose these points sufficiently close to $\mathbf{q}$ that the line segment $\overline{\mathbf{p}'\mathbf{p}}$ is covered by d-simplices containing $\mathbf{q}$. If necessary, we perturb the points such that $\overline{\mathbf{p}'\mathbf{p}}$ does not intersect any subsimplex of T of dimension less than $d-1$. At some point between $\mathbf{p}'$ and $\mathbf{p}$, the line segment must pass from an intersecting into a non-intersecting d-simplex. This point is interior to a facet $\mathbf{f}$, which is shared by the two d-simplices. Since the non-intersecting d-simplex contains $\mathbf{q}$, it is a closest non-intersecting d-simplex. ∎

The algorithm starts by inserting one d-simplex incident on $\mathbf{p}_i$ into the empty heap. It then keeps processing simplices from the top of the heap until it has found the k nearest neighbors. If the simplex from the heap is a vertex, it is reported as the next neighbor. When a d-simplex $\mathbf{t}$ is taken from the heap, it becomes intersecting. The d adjacent d-simplices and the vertices of $\mathbf{t}$ are inserted into the heap. A flag for each d-simplex and each vertex prevents multiple insertion into the heap. The flag is set when its corresponding simplex is inserted. A simplex whose flag is set will not be inserted again.

As mentioned above, the heap is ordered by distance from the query point. As a secondary ordering criterion, vertices are given priority over d-simplices: If a vertex and a d-simplex are equally distant from $\mathbf{p}_i$, the vertex will appear at the top of the heap first. This prevents the algorithm from unnecessarily processing d-simplices which are as far from $\mathbf{p}_i$ as the k^{th} nearest neighbor. The complete algorithm is described in Algorithm B.1.

Algorithm B.1 k-Nearest-Neighbors Query

```
Input: Triangulation T of a point set P = {p_1, ..., p_n}, query point p_i ∈ P, and integer k.
Operation: Compute k nearest neighbor points of p_i :
Heap H := ∅.
j := 0.    // number of neighbors found so far
Find a d-simplex t which is incident to p_i .
Insert t into H.
Set the flag of t.
repeat
  Delete simplex s from the top of H.    // we now call s intersecting
  if ( s is a d-simplex ) then
    foreach ( vertex v of s with flag of v not set ) do
      Compute d(v, p_i).
      Insert v into H.
      Set the flag of v.
    end
    foreach ( d-simplex t adjacent to s with flag of t not set ) do
      Compute d(t, p_i).
      Insert t into H.
      Set the flag of t.
    end
  else  // s is a vertex
    j := j + 1.
    Report s as the j^th neighbor.
  end
until ( j = k )
Output: k nearest neighbor points of p_i.
```

Theorem B.4 *Algorithm B.1 reports k nearest neighbors of* $\mathbf{p}_i$ *in order of increasing distance.*

Proof: Let us first consider the case $k = n - 1$, i.e., all other vertices are requested. In this case, we have to show that the vertices are reported in the correct order. Assume that vertex $\mathbf{q}$ is reported before $\mathbf{p}$, but $\mathbf{p}$ is strictly closer to $\mathbf{p}_i$ than $\mathbf{q}$. This can only happen if $\mathbf{q}$ appears at the top of the heap before $\mathbf{p}$ has been inserted. There exists a d-simplex $\mathbf{t}$ which is incident on $\mathbf{p}$. Now $\mathbf{t}$ must be non-intersecting, or $\mathbf{p}$ would have been inserted into the heap. On the other hand, $\mathbf{t}$ is not further from $\mathbf{p}_i$ than $\mathbf{p}$, and therefore strictly closer than $\mathbf{q}$. By Lemma B.3, there exists a closest non-intersecting d-simplex $\mathbf{t}'$ which is adjacent to an intersecting d-simplex. Because of this adjacency, $\mathbf{t}'$ must be in the heap. On the other hand, $\mathbf{t}'$ is closer than $\mathbf{q}$, a contradiction. Therefore, $\mathbf{q}$ cannot be reported before $\mathbf{p}$. To prove the case $k < n - 1$, we simply note that the algorithm runs in exactly the same way as for $n - 1$ neighbors, but stops after the k nearest neighbors have been found. ∎

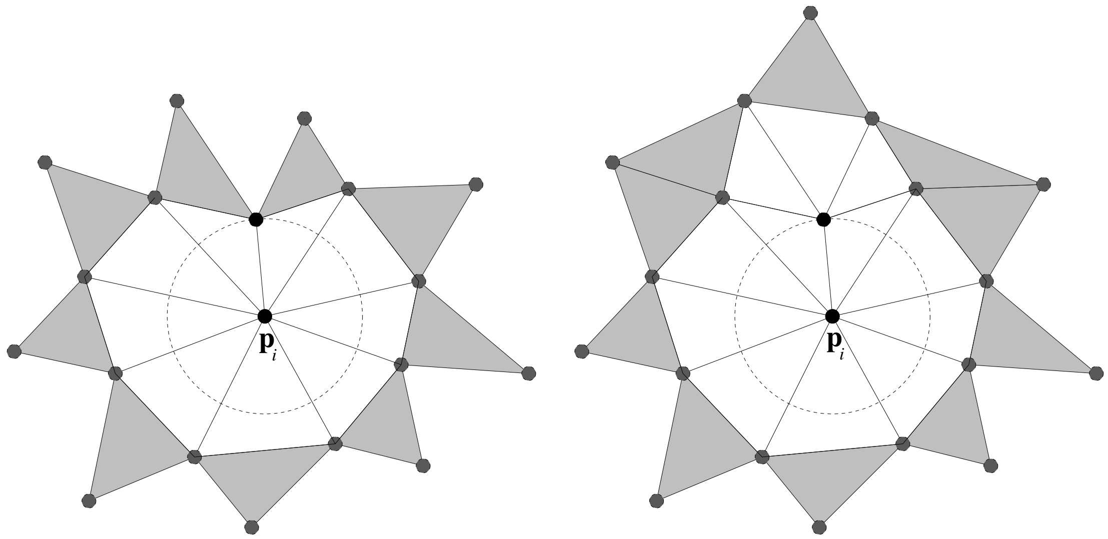


Figure B.1: Two phases of the algorithm as the expanding ball encounters a vertex and, simultaneously, three triangles.

Figure B.1 shows two snapshots of the algorithm working on a planar triangulation. The intersecting triangles and vertices are drawn in white and black, respectively. The triangles and vertices in the heap are drawn in gray. In the left diagram, the algorithm has just deleted a vertex from the heap. In the right diagram, it has also processed the triangles that are incident on this vertex.

Concerning the analysis of the time complexity of the algorithm, we assume that the data structure of the triangulation allows us to carry out the following operations:

- Given a vertex, find an incident d-simplex in constant time.
- Given a d-simplex, find its vertices in time $\mathcal{O}(d)$.
- Given a d-simplex, find the $d + 1$ adjacent d-simplices in time $\mathcal{O}(d)$.

One elementary step in Algorithm B.1 is the distance computation between a d-simplex and the query point. The time complexity of this step depends on the dimension and on the metric being used. In d-dimensional space, it takes time proportional to d to determine the Euclidean distance between two points alone. In the following, we let δ denote the worst-case complexity of distance computations, both between two points and between a point and a d-simplex.

Let $|T|$ denote the number of d-simplices in T, and let $|H|$ be the number of simplices contained in H. We note that $n \in \mathcal{O}(|T|)$. In the planar case, we also have $|T| \in \mathcal{O}(n)$. In 3-dimensional space, triangulations with $|T| \in \mathcal{O}(n)$ exist and can be constructed in $\mathcal{O}(n \log n)$ time (cf. [EPW90]).

To analyze the time complexity of the overall algorithm, let us first look at the time spent on heap operations. An insertion or deletion takes $\mathcal{O}(\log |H|)$ time. Each vertex and each d-simplex is inserted into, and, likewise, deleted from, the heap at most once. Therefore, both the number of heap operations and the heap size are bounded by $\mathcal{O}(|T|)$. The total time for all heap operations is $\mathcal{O}(|T| \log |T|)$ in the worst case. Distance computations are carried out only for those simplices which are inserted into the heap, and only once per simplex. Therefore, the total time for distance computations is bounded by $\mathcal{O}(\delta\,|T|)$.

Next, we will look at the two for-loops. Disregarding the heap operations and distance computations, for which we have already accounted above, the body of each for-loop consists only of setting a flag. This can be done in constant time. Within one execution of the repeat-loop, each for-loop is run at most $d+1$ times. The loop overhead consists of finding $d+1$ vertices or d-simplices and testing their flags, which takes time proportional to d. Thus, the for-loops cost $\mathcal{O}(d)$ time. All other steps that we have not considered so far require constant time. Each time the repeat-loop is executed, a simplex is deleted from the heap. This bounds the number of executions of the repeat-loop with $\mathcal{O}(|T|)$. Thus, the time for all executions of all constant-time steps is bounded by $\mathcal{O}(d\,|T|)$. This results in a total execution time of $\mathcal{O}(|T|\,(d+\delta+\log|T|))$ in the worst case.

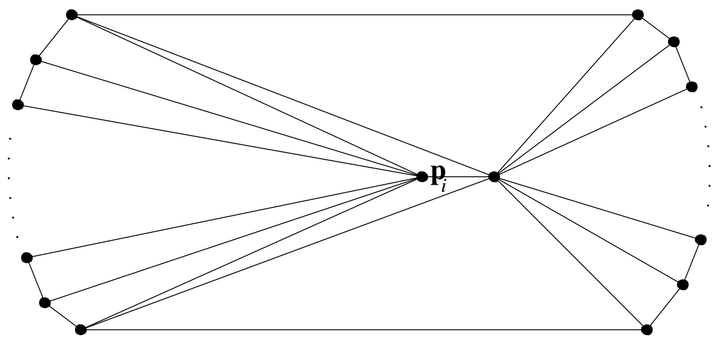


Figure B.2: Triangulation causing worst-case behavior of the algorithm.

Figure B.2 shows a planar example which causes worst-case behavior of the algorithm. The dots in the diagram indicate that the left and right boundaries have $\frac{n}{2}-1$ vertices each, where n may be arbitrarily large. We consider a query with $k=2$ at the time when the nearest neighbor of $\mathbf{p}_i$ has just become intersecting. The heap contains the $\frac{n}{2}-1$ vertices left of $\mathbf{p}_i$. Before the next neighbor can be found, $\frac{n}{2}-2$ triangles and $\frac{n}{2}-1$ vertices lying to the right of the nearest neighbor are inserted. Since these new simplices are closer to $\mathbf{p}_i$ than the $\frac{n}{2}-1$ vertices already contained in the heap, the summed cost for the insertions is proportional to $n\log n$. Note that this extreme behavior of the algorithm occurs only if $\mathbf{p}_i$ or its nearest neighbor is used as the query point. For any other vertex, at most 6 triangles and 7 vertices are inserted into the heap before the second neighbor is found. In fact, as long as k is small compared to n, the time complexity averaged over all vertices depends on k rather than on $|T|$. We conjecture that dependency on k only will be the case for most triangulations. This conjecture is strongly supported by the experimental results which are presented next.

In order to investigate the algorithm's behavior in practice, it was measured on various point sets. The experiments were set up as follows. For each point $\mathbf{p}_i$ in a data set P, a query for the 2000 nearest neighbors of $\mathbf{p}_i$ is carried out. When the query finds the j^{th} neighbor, $1\le j\le 2000$, two quantities are recorded: the current heap size, denoted by $|H|\,(\mathbf{p}_i,j)$, and the number of heap insertions which the query has executed up to this point, denoted by $\#I(\mathbf{p}_i,j)$. Note that these quantities reflect not only the current state of the actual query for 2000 neighbors, but also the final state of a hypothetical query for only j neighbors of $\mathbf{p}_i$.

The measurements were carried out on eight two-dimensional data examples:

- three sets of uniformly distributed random points, containing 2500, 10000, and 100000 points,
- three square grids of sizes 50×50, 100×100, and 200×500, and
- two sets of 8700 and 13687 points, scanned from real objects and exhibiting strong variation in point density due to previous data reduction.

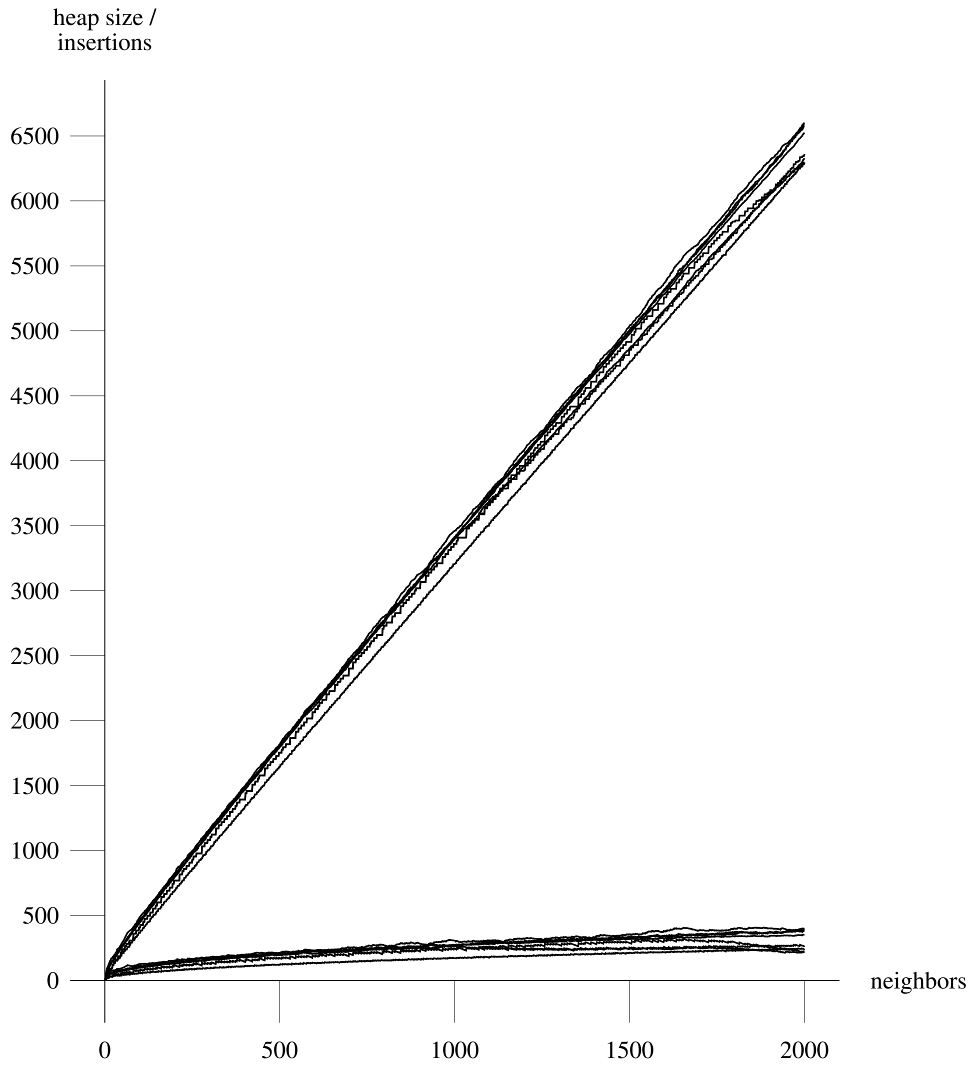


Figure B.3: Number of heap insertions (upper set of curves) and current heap size (lower set of curves).

In each case, T was a Delaunay triangulation of P.

We were interested in the worst-case behavior of the algorithm on each particular data set, so the maxima

$$|H|(j) := \max_{\mathbf{p}_i \in P} |H|(\mathbf{p}_i, j) \quad \text{and} \quad \#I(j) := \max_{\mathbf{p}_i \in P} \#I(\mathbf{p}_i, j)$$

over all queries within the same data set P were computed. Figure B.3 shows the graphs of $|H|(j)$ and $\#I(j)$ as functions of j. Two observations can be made in the graphs. The first is that $\#I(j)$ is strongly correlated to j. In other words, it appears to depend on j linearly. The second observation is that, although the underlying data sets vary in size by a factor of up to 40, the corresponding function graphs in Figure B.3 almost coincide. This indicates that, as far as our examples go, the time complexity is in fact independent of $|T|$.

In some applications, one does not know a priori how many nearest neighbors of a query point will be required. After looking at the k nearest neighbors, one may find that another ℓ neighbors are necessary. In such a situation, it is an easy matter for Algorithm B.1 to resume the query where it left off before.

The cost for searching first k and then the next ℓ neighbors is the same as for searching $k+\ell$ neighbors in a single query. Some information must be saved in order to resume a query. This comprises the heap H, the flags, and the number k of previously found neighbors. We call this information *nearest neighbor query iterator* (*NNQ iterator*).
It is also possible to run queries in a concurrent manner, e.g., find k_i neighbors of $\mathbf{p}_i$, then k_j neighbors of $\mathbf{p}_j$, then another ℓ_i neighbors of $\mathbf{p}_i$, etc. Multiple suspended queries require a nearest neighbor iterator for each query point. Suppose that concurrent queries are carried out for all n vertices, then the space requirement for the flags is proportional to $n\,|T|$. However, if the average number of neighbors computed per vertex is small, most of the flags will never be used. Storage space can be reduced if we replace the flags by a hash table. Instead of setting a flag, we insert a pair of the form (query point , 'flagged' simplex) into the table. By the very nature of concurrent queries, the required number of neighbors in a single query is not known in advance. Thus, it may be impossible to make an appropriate choice for the size of the hash table, which has a strong influence on the table's efficiency. As an alternative, we can substitute a sorted tree (e.g., AVL or SBB tree, cf. [Wir86]) for each heap, i.e., one per query. We insert $\mathbf{s}'$ into the tree if it is not contained in the tree and was not processed before $\mathbf{s}$. Containment in the tree can be tested efficiently. Simplices are processed in order of increasing distance from the query point. Therefore, if $\mathbf{s}'$ is strictly closer than $\mathbf{s}$, it has been processed before, and if $\mathbf{s}'$ is strictly further than $\mathbf{s}$, it has not been processed before. Equidistant simplices that have been processed are stored in an auxiliary tree, and can be found there. A simplex is inserted into the auxiliary tree as soon as it is processed. The auxiliary tree is cleared when the intersecting sphere expands, i.e., when a simplex of greater distance is processed.

B.2 An Object-Oriented Framework for Flexible and Adaptive Nearest Neighbor Queries

Usually, nearest neighbor queries are only considered from the algorithmical point of view and not from the implementation view. Since the nearest neighbor task is a frequent problem in many applications we show how the implementation can be done with respect to our algorithmical solution. The reason for this is, that our approach is so general that it allows an easy-to-follow object-oriented approach that simplifies the implementation. Another advantage is that this framework can be adjusted to several similar tasks of nearest neighbor queries, for example, if using other types of spatial subdivisions besides tetrahedrizations/triangulations.
In the following we describe the basic approach how the implementation can be done. In our example we restrict the types of query objects to single points and the type of spatial subdivisions to tetrahedrizations as examined in theory for the k nearest neighbor search in the previous section. But extension to arbitrary types of objects (polyhedrons) is straightforward. The only thing that has to be considered is an appropriate distance function for the kind of object that is used as query object and the surrounding polyhedral cells. The already reported neighbors are stored in a list that is associated with the query point so that concurrent queries that are carried out from different positions of the program do not repeat unnecessary computations. This means, that if the number of needed neighbors is below the number of already computed neighbors, only the elements of the neighbor list of a query point have to be iterated in order to complete the query task. This holds as long as the tetrahedrization is not modified. But even if this is the case, local modifications mean only local influence on the neighbor structure for the points in the modified regions, so that an update can be established efficiently by only updating the points that have already "traversed" the modified region with their expanding ball.
For better understanding of the implementation be have chosen a simplified and more readable version of C++ – like program code. This code should be comprehensible for everyone who is familiar with object-oriented concepts in an arbitrary programming language.

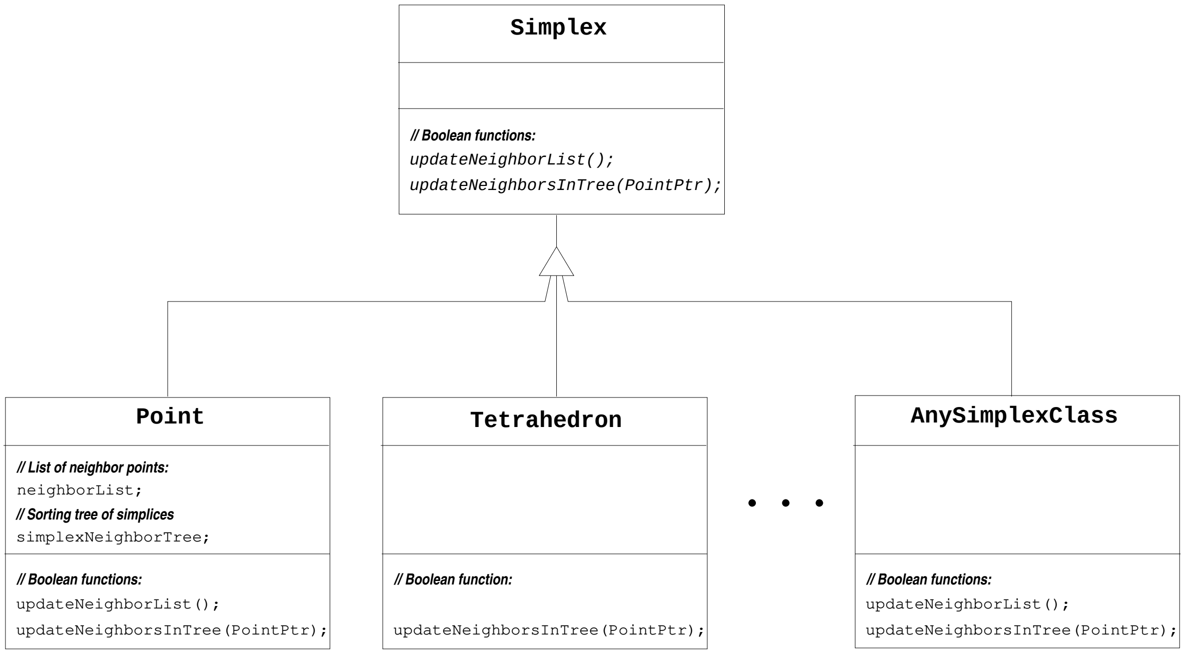


Figure B.4: The UML class diagram of the nearest neighbor query implementation.

We assume an arbitrary tetrahedrization is already given and use this information within the nearest neighbor query algorithm.

The *UML class diagram (unified modeling language)* [BRJ99] of our implementation is shown in Figure B.4. In our application we need a `Point` class and a `Tetrahedron` class that are both derived from the class `Simplex`. Since this concept is extensible to any type of nearest neighbor queries for each kind of simplex, there also can be implemented a nearest neighbor query for an arbitrary simplex class which is denoted in the diagram as `AnySimplexClass`. The most important parts of the diagram are the *virtual functions* `updateNeighborList()` and `updateNeighborsInTree()` of class `Simplex` that are overloaded in each implementation inside the class `Point` or `Tetrahedron`. These are the main functions needed in the realization of a k-nearest-neighbor query algorithm. As we see in the diagram the function `updateNeighborList()` is only implemented for the `Point` class, since we restrict our queries to points. But as mentioned before, arbitrary queries for other object types than points would be possible, too. For example, if one would like to implement a nearest neighbor query for tetrahedrons, it is sufficient to implement the function `updateNeighborList()` for the `Tetrahedron` class.

In the following we describe how the concept of the UML class diagram can be used in order to implement nearest neighbor queries for points. Class `NnqIterator` delivers the frame work for an instantiation of a nearest neighbor query iterator for a query point **q**, cf. Figure B.5. It consists of a pointer to a point that describes the query object, and a list iterator that iterates on the list of nearest neighbors already found. This `neighborList` is associated with each query point. In the beginning the `neighborList` is empty before a query has taken place for the query point. With the `reset()` function of `NnqIterator` the iterator can be set to the beginning of a neighbor query which responds to the situation where the query ball has radius zero. In order to get a nearest neighbor the `get()` function of `NnqIterator` is then called. Then the next point that is delivered is the nearest neighbor of the query. Each time the `get()` function is called a new neighbor is computed and appended to the `neighborList` that is associated with the query point **q**. But first it is checked

```
class NnqIterator
{
  PointPtr                    q;         // pointer to query point
  ListIterator<PointPtr>      listIter;  // iterator for ordered nearest neighbor list of already computed neighbors
public:
  :
  void     reset() { listIter.reset(); }  // reset list iterator; query ball radius = 0
  Boolean  get(PointPtr & p)              // compute next nearest neighbor
           {
             if ( listIter.get(p) )        // nearest neighbor already computed ?
                      // Yes, return true and reference to element p
                       { return TRUE; }
                      // No, then compute next nearest neighbor
             else  {
                       q→updateNeighborList(); // compute next neighbor
                       // now, the neighbor list for the point on which listIter iterates is updated
                       // if a new element has been added to the list,
                       // then the next call of listIter.get(p) returns the next neighbor
                       // if a new neighbor has been found
                       return listIter.get(p);  // return next neighbor
                     }
           }
}
```

Figure B.5: The listing of the *nearest neighbor query iterator* class: NnqIterator.

for whether there is already a next nearest neighbor in the neighborList that has not been reported so far. This is performed by calling the get() function of the list iterator with listIter.get(). If a neighbor is on a list the algorithm returns the value TRUE and a reference to the neighbor point found. If this is not the case then the next nearest neighbor is computed by calling the function **q**→updateNeighborList() of the query point. This function starts the complete nearest neighbor computation process. If a new neighbor could be found, it is appended to the neighborList, so that the next call of the listIter.get() function of the list iterator will return an element and the value TRUE.

Now we come to the description of an update of the neighborList of a query point, cf. Figure B.6. As described in the UML diagram the class Point has two important member functions: updateNeighborList() and updateNeighborsInTree(). The function updateNeighborList() is needed for the initiation of a query while the relevant update procedures are called with updateNeighborsInTree(). Since we do not show here how nearest neighbor queries can be applied to tetrahedra the class Tetrahedron needs only to overload the function updateNeighborsInTree() so that neighbor queries for points can be applied. The function **q**→updateNeighborList() returns TRUE if a new neighbor point could be found. This is only the case if the recursive call of updateNeighborsInTree() for each simplex on **s** during the *neighbor update process* results in finding a new neighbor point.

When a neighbor list update for a query point **q** is applied, the function **q**→updateNeighborList() is called. If the neighborList is empty, the neighbor update for **q** must be intialized by putting all adjacent tetrahedra **t** to the tree of simplices. These simplices are stored in the simplexNeighborTree that contains all points and tetrahedra that surround the "query ball". These elements are sorted according to their distance to the query point **q**. This is done in the **foreach**-loop of Figure

```
Boolean Point::updateNeighborList()
{
 if ( neighborList.isEmpty() )    // initialize the neighbor structure around this point
   {
    Tetrahedron t;
    foreach ( adjacent tetrahedron t of this point instance ) do
       {
        putToSimplexNeighborTree(t,dist(t,this)); // put tupel (t,dist(t,this))
                                                  // on the sorted neighbor simplex tree
        t→putToVisitTree(this);                   // mark tetrahedron t as visited from this point
       }
   }
 Simplex s;
 if ( getFromSimplexNeighborTree(s) )
      { return s→updateNeighborsInTree(this); }
 else { return FALSE;                           }
}
```

Figure B.6: The listing of the function `updateNeighborList` of the `Point` class.

B.6.

If the `neighborList` has been already initialized, then the next nearest neighbor simplex **s** (point or tetrahedron) of the query point is taken by `getFromSimplexNeighbor(s)`. Then, the function **s**→`updateNeighborsInTree()` of this simplex **s** is called.

Now, two cases can occur. Either **s** is a point or a tetrahedron.

We first consider the case, that the simplex **s** is a point. In its associated function **s**→`updateNeighborsInTree()` we first memorize that **s** has been visited by **q** by calling the function `putToVisitTree()`, cf. the listing in Figure B.7. Then, the point is appended to the `neighborList` of the query point.

```
Boolean Point::updateNeighborsInTree(PointPtr q)
//
// member function call for this instance of class Point
//
{
  // mark this instance p_this of class Point as visited from point q
  this→putToVisitTree(q);          // memorize query q in a AVL tree that this point p_this was visited by q
  q→appendToNeighborList(this);    // append pointer to this instance p_this of class Point
                                   // to the neighbor list of q
  return TRUE;
}
```

Figure B.7: The listing of the function `updateNeighborsInTree` of the `Point` class.

If the simplex **s** is a tetrahedron then its associated function **s** →`updateNeighborsInTree()` does the following, cf. Figure B.8. First, all four points of the tetrahedron are marked as visited from the query point **q**. Second, the four points are put onto the simplex neighbor tree of the query point **q** which sorts the simplices according to their distance to **q** in shortest first ordering. Then, all tetrahedra that have not been visited by the query point **q** are put onto the simplex neighbor tree that sorts all simplices. Additionally, each tetrahedron that is considered in this previous step is marked as being already visited by **q** so that it is not considered again. All this is performed in the **foreach**-loop. After that, the next nearest neighbor simplex (either a point or a tetrahedron) is taken from the simplex neighbor tree and its `updateNeighborsInTree()` function is called.

This process of calling the function **s**→`updateNeighborsInTree()` for each simplex **s** iterates on the `simplexNeighborTree` until the simplex **s** is a point so that it can be added to the `neighborList` as new nearest neighbor. If no point is found, than the whole point set has been already traversed by the query for **q** and the process terminates. In this case the iterator for the `neighborList` does not return a new neighbor. Otherwise, if a new neighbor has been found the `neighborList` increases in length by one element and the list iterator in class `NnqIterator` returns a new element if the `listIter.get()` function is called.

```
Boolean Tetrahedron::updateNeighborsInTree(PointPtr q)
//
// this is the member function call for tetrahedron this
//
{
 // 1. mark all four points a,b,c,d of the tetrahedron this=◇(a,b,c,d)
 //    as visited from the query point q.
 // 2. put all four points a,b,c,d on the simplex neighbor tree of q
 //    sorted according to their distance to q (shortest first).
 .
 .   ⟵    /* here is the relevant program code for the two steps above */
 .
 // put all tetrahedra that have not been visited by q onto the simplex neighbor tree
 foreach ( adjacent tetrahedron t of this tetrahedron instance ) do
 {
    if ( t→isNotContainedInVisitTree(q) )
       {
        t→putToVisitTree(q);
        q→putToSimplexNeighborTree(t,dist(t,q));
       }
 }
 // get next nearest neighbor simplex (point or tetrahedron) for point q
 if ( q→getFromSimplexNeighborTree(s) )
       { return s→updateNeighborsInTree(q); }
 else  { return FALSE;                      }
}
```

Figure B.8: The listing of the function `updateNeighborsInTree` of the `Tetrahedron` class.

An example for the usage of this program framework is given in Example B.5.

Example B.5 (Usage of the k-nearest-neighbors query implementation) *Let* **q** *be the query point. Then, the k nearest neighbors can be computed very simply:*

```
NnqIterator iter(q);
Point       nn;
Number      i=1;
Number      k=11;  // number of nearest neighbors to be computed

while ( (i ≤ k) AND (iter.get(nn)) ) // print neighbors in order of appearence
{
  cout << "Neighbor number " << i << ":  " << nn→pointIndex() << endl;
}
```

If a new `NnqIterator` *for the same* **q** *is instantiated during the program execution in order to compute* r *nearest neighbors then we have two possible cases. If* $r \leq k$*, it means that the neighbors to be delivered are taken from an already computed list of nearest neighbors. Here, no more update with tetrahedral sorting and searching in the above mentioned data structures is necessary.*

```
       i=1;
Number r=8;  // number of nearest neighbors to be computed

while ( (i ≤ r) AND (iter.get(nn)) )  // print neighbors in order of appearence
{
  cout << "Neighbor number " << i << ":  " << nn→pointIndex() << endl;
}
```

If $r > k$*, it means that the already* k *computed neighbors are taken and that for the last* $r-k$ *neighbors a new nearest neighbor computation is started by calling the function* **q**→`updateNeighborList()`.

```
.
·    // as before
.
Number r=17;  // number of nearest neighbors to be computed

while ( (i ≤ r) AND (iter.get(nn)) )  // print neighbors in order of appearence
{
  cout << "Neighbor number " << i << ":  " << nn→pointIndex() << endl;
}
```

After r *nearest neighbors are computed, then* $r = 17$ *elements are part of the nearest neighbor list that is associated with the specific query point* **q**.

Bibliography

[AB98] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. In *14th ACM Symposium on Computational Geometry*, pages 39–48, 1998.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi–based surface reconstruction algorithm. *Computer Graphics*, pages 415–421, 1998. Proceedings of SIGGRAPH '98.

[AC99] N. Amenta and S. Choi. One-pass Delaunay filtering for homeomorphic 3D surface reconstruction. Technical Report TR99-08, University of Texas, Austin, 1999.

[ACDL00] N. Amenta, S. Choi, T.K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of 16th ACM Sympos. Comput. Geom.*, July 2000.

[ACK01] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19(2-3):127–153, 2001.

[AGJ00] U. Adamy, J. Giesen, and M. John. New techniques for topologically correct surface reconstruction. In *Proceedings of IEEE Visualization 2000*. IEEE Computer Society Press, 2000.

[AGJ01] U. Adamy, J. Giesen, and M. John. Surface reconstruction using umbrella filters. *Computational Geometry: Theory and Applications*, 2001.

[AM91] S. Abramowski and H. Müller. *Geometrisches Modellieren.* BI–Wiss.–Verl., 1991. Bd. 75.

[AMN$^+$94] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest–neighbour searching. In *Proceedings 11th Annual Symposium on Discrete Algorithms*, pages 573–582, 1994.

[AS96] M.-E. Algorri and F. Schmitt. Surface reconstruction from unstructured 3D data. *Computer Graphics Forum*, 15(1):47–60, 1996.

[AS00] M. Attene and M. Spagnuolo. Automatic surface reconstruction from point sets in space. *Computer Graphics Forum*, 19(3), 2000. Proceedings of EUROGRAPHICS 2000.

[Att97] D. Attali. r-regular shape reconstruction from unorganized points. In *ACM Symposium on Computational Geometry*, pages 248–253, 1997. Nice, France.

[Baa95] A. Baader. *Ein Umwelterfassungssystem für multisensorielle Montageroboter.* PhD thesis, Universität der Bundeswehr, Munich, Germany, 1995. Fortschrittberichte, VDI Reihe 8 Nr. 486, ISBN 3-18-3 48608 - 3, ISSN 0178-9546.

[BB97] F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. In *Proc. of the Ninth Canadian Conference on Computational Geometry*, pages 193–198, August 1997. Also available as Technical Report CSD-97-013, Department of Computer Sciences, Purdue University.

[BBCS97] F. Bernardini, C. Bajaj, J. Chen, and D.R. Schikore. Automatic reconstruction of 3D CAD models from digital scans., 1997. Technical Report CSD-97-012, Department of Computer Sciences, Purdue University.

[BBX95] C.L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. *Computer Graphics Proceedings, SIGGRAPH '95, Annual Conference Series*, pages 109–118, 1995.

[BBX97] C. Bajaj, F. Bernardini, and G. Xu. Reconstructing surfaces and functions on surfaces from unorganized 3D data. *Algorithmica*, 19:243–261, 1997.

[Ben74] C. Bender. Bestimmung der grössten Anzahl gleicher Kugeln welche sich auf eine Kugel von demselben Radius, wie die übrigen, auflegen lassen. *Archiv Math. Physik (Grunert)*, 56:302–306, 1874.

[Ber63] C. Berge. *Topological Spaces*. Oliver & Boyd Ltd., Edinburgh and London, 1963.

[Ber96] F. Bernardini. *Automatic Reconstruction of CAD Models and Properties from Digital Scans*. PhD thesis, Department of Computer Science, Purdue University, 1996.

[BH93] A. Baader and G. Hirzinger. Three–dimensional surface reconstruction based on a self–organizing feature map. In *Proc. 6th Int. Conf. Advan. Robotics*, pages 273–278, 1993. Tokyo.

[BH94] A. Baader and G. Hirzinger. A self–organizing algorithm for multisensory surface reconstruction. In *International Conf. on Robotics and Intelligent Systems IROS '94*, September 1994. Munich, Germany.

[BJMT99] F. Bernardini, C. Silva J. Mittleman, H. Rushmeier, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4), 1999.

[Blu97] S. Blumenthal. Moderne Methoden der Mensch-Maschine-Interaktion im Geometrischen Modellieren. Master's thesis, Informatik VII (Computer Graphics), University of Dortmund, Germany, 1997.

[Boe52] A. H. Boerdijk. Some remarks concerning close-packing of equal size. *Philips Res. Rep.*, 7:303–313, 1952.

[Boi84] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, October 1984.

[Bor26] O. Borůvka. O jistém problému minimálním. *Práce moravské přírodovědecké společnosti*, 3:37–58, 1926. in Czech.

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[Bro91] J.L. Brown. Vertex based data dependent triangulations. *Computer Aided Geometric Design*, 8:239–251, 1991.

[BS87] I.A. Bronstein and K.A. Semendjajew. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 1987.

[BS91] G. Baszenski and L.L. Schumaker. Use of simulated annealing to construct triangular facet surfaces. In *Curves and Surfaces*, pages 27–32. Academic Press, Boston, 1991.

[BS96] C. Bajaj and D. Schikore. Error-bounded reduction of triangle meshes with multivariate data. In *Proceedings of SPIE Symposium on Visual Data Exploration and Analysis III*, pages 34–45, January 1996. SPIE.

[BS97] R.E. Bank and R.K. Smith. Mesh smoothing using a posterior error estimation. *SIAM Journal on Numerical Analysis*, 1997.

[BTG95] E. Bittar, N. Tsingos, and M.-P. Gascuel. Automatic reconstruction of unstructured data: Combining a medial axis and implicit surfaces. *Computer Graphics Forum*, 14(3):457–468, 1995. Proceedings of EUROGRAPHICS '95.

[CGA] Computational geometry algorithms library. `http://www.cgal.org`.

[CK92] P.B. Callahan and S.R. Kosaraju. A decomposition of multi-dimensional point-sets with applications to k-nearest-neighbours and n-body potential fields. In *Proceedings 24th Annual ACM Symposium on the Theory of Computing*, pages 546–556, 1992.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Computer Graphics Proceedings, SIGGRAPH '96, Annual conference series*, pages 303–312, 1996.

[CS88] J.H. Conway and N.J.A. Sloane. *Sphere Packings, Lattices and Groups*. Springer Verlag, 1988.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley and Sons, Inc., 1973.

[DLR90] N. Dyn, D. Levin, and S. Rippa. Algorithms for the construction of data dependant triangulations. In *Algorithms for Approximation II*, pages 185–192. Chapman and Hall, London, 1990.

[Ede87] H. Edelsbrunner. *Algorithms in Computational Geometry*. Springer Verlag, 1987.

[Ede92] H. Edelsbrunner. Weighted alpha shapes, 1992. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[EH96] M. Eck and H. Hoppe. Automatic reconstruction of B–spline surfaces of arbitrary toplogical type. *Computer Graphics Proceedings, SIGGRAPH '96, Annual Conference Series*, pages 325 – 334, 1996.

[EM94] H. Edelsbrunner and E. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994. Also as Technical Report UIUCDCS-R-92-1734, Department of Computer Science, 1992, University of Illinois at Urbana-Champaign.

[EPW90] H. Edelsbrunner, F.P. Preparata, and D.B. West. Tetrahedrizing point sets in three dimensions. *Journal of Symbolic Computation*, 10:335–347, 1990.

[FCGA97] E. Ferley, M.-P. Cani-Gascuel, and D. Attali. Skeletal reconstruction of branching shapes. *Computer Graphics Forum*, 16(5):283–293, December 1997.

[FOG97] L. Freitag and C. Ollovier-Gooch. Tetrahedral mesh improvement using face swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40, 1997.

[FS91] P. Fua and P.T. Sander. From points to surfaces. In Baba C. Vemuri, editor, *Geometric Methods in Computer Vision*, pages 286–296. Proc. SPIE Vol. 1570, 1991.

[FS92a] P. Fua and P.T. Sander. Reconstructing surfaces from unstructured 3D points. In *Proc. Image Understanding Workshop*, pages 615–625, San Diego, CA, 1992.

[FS92b] P. Fua and P.T. Sander. Segmenting unstructured 3D points into surfaces. In G. Sandini, editor, *Computer Vision : ECCV '92, Proc. Second European Conference on Computer Vision*, pages 676–680, Santa Margherita Ligure, Italy, May 1992. Springer.

[GKS00] M. Gopi, S. Krishnan, and C.T. Silva. Surface reconstruction based on lower dimensional localized Delaunay triangulations. *Computer Graphics Forum*, 19(3), 2000. Proceedings of EUROGRAPHICS 2000.

[GMW97] B. Guo, J. Menon, and B. Willette. Surface reconstruction using alpha–shapes. *Computer Graphics Forum*, Vol. 16(No. 4):177–190, October 1997.

[GS69] K.R. Gabriel and R.R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[Gün75] S. Günther. Ein stereometrisches Problem. *Archiv Math. Physik (Grunert)*, 57:209–215, 1875.

[Guo97] B. Guo. Surface reconstruction: from points to splines. *Computer-Aided Design*, Vol. 29(No. 4):269–277, April 1997.

[Ham97] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, 1997.

[HDD$^+$92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, July 1992. Proceedings of SIGGRAPH '92.

[HDD$^+$93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Computer Graphics Proceedings*, Annual Conference Series, pages 21–26, New York, 1993. ACM Siggraph. Proceedings of SIGGRAPH '93.

[Hei98] B. Heinz. Entwurf und Realisierung eines integrierbaren Regelevaluierungssystems für geometrische und topologische Anfragen bei der strukturellen Analyse von Daten. Master's thesis, Informatik VII (Computer Graphics), University of Dortmund, Germany, 1998.

[HL79] G.T. Herman and H.K. Liu. Three-dimensional displays of human organs from computed tomograms. *Computer Graphics and Image Processing*, 9:1–21, January 1979.

[HL92] J. Hoschek and D. Lasser. *Grundlagen der geometrischen Datenverarbeitung.* B.G. Teubner, Stuttgart, 1992.

[HL93] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A.K. Peters, 1993.

[Hop74] R. Hoppe. Bemerkung der Redaction (supplementary notes to the article of C. Bender on the pages 302-306 in the same issue). *Archiv Math. Physik (Grunert)*, 56:307–312, 1874.

[Hop94] H. Hoppe. *Surface Reconstruction from Unorganized Points*. PhD thesis, Univ. of Washington, Seattle WA, 1994.

[IBS97] F. Isselhard, G. Brunnett, and T. Schreiber. Polyhedral reconstruction of 3D objects by tetrahedra removal. Technical report, Fachbereich Informatik, University of Kaiserslautern, Germany, February 1997. Internal Report No. 288/97.

[Joe89] B. Joe. Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comput.*, 10(4):718–741, July 1989.

[Joe91] B. Joe. Construction of three-dimensional Delaunay triangulations using local transformations. *Computer Aided Geometric Design*, 8:123–142, 1991.

[KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multiresolution modeling on arbitrary meshes. *Computer Graphics*, pages 105–114, 1998. Proceedings of SIGGRAPH '98.

[Koh99] M. Kohler. *New Contributions to Vision-Based Human-Computer Interaction in Local and Global Environments*. PhD thesis, Informatik VII (Computer Graphics), University of Dortmund, Germany, 1999.

[KR85] D.G. Kirkpatrick and J.D. Radke. A framework for computational morphology. In *Computational Geometry*, pages 217–248. Elsevier Science Publishers B.V. (North-Holland), 1985.

[Kur68] K. Kuratowski. *Topology (Volume II).* Academic Press, 1968.

[Lam93] I.K. Lam. *Tix Programmer's Guide*, 1993. `http://tix.sourceforge.net`.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[Lee56] J. Leech. The problem of thirteen spheres. *Math. Gazette*, 40:22–23, 1956.

[Mai98] M. Maier. Merkmalserkennung und -verarbeitung in Punktmengen und Flächenbeschreibungsgraphen. Master's thesis, Informatik VII (Computer Graphics), University of Dortmund, Germany, 1998.

[MBL+91] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, and M.J. Wozny. Geometrically deformed models: A method for extracting closed geometric models from volume data. *Computer Graphics*, pages 217–226, July 1991. Proceedings of SIGGRAPH '91.

[Men95] R. Mencl. A graph–based approach to surface reconstruction. *Computer Graphics Forum*, 14(3):445–456, 1995. Proceedings of EUROGRAPHICS '95, Maastricht, The Netherlands.

[MK93] H. Müller and A. Klingert. Surface interpolation from cross sections. In H. Hagen, H. Müller, and G.M. Nielson, editors, *Focus on Scientific Visualization*, pages 139–189. Springer Verlag, 1993.

[MM98a] R. Mencl and H. Müller. Graph–based surface reconstruction using structures in scattered point sets. In *Proceedings of CGI '98 (Computer Graphics International)*, pages 298–311. IEEE Computer Society Press, 1998.

[MM98b] R. Mencl and H. Müller. Interpolation and approximation of surfaces from three–dimensional scattered data points. In *State of the Art Reports (STAR) of EUROGRAPHICS '98*, pages 51–67, 1998. Lisbon, Portugal.

[Müc93a] E.P. Mücke, 1993. Manual for the 3D Delaunay triangulation program `detri`, `(detri/README.tex)`, theory and practice.

[Müc93b] E.P. Mücke. *Shapes and implementations in three–dimensional geometry*. PhD thesis, Department of Computer Science, University of Illinois at Urbana–Champaign, September 1993.

[Mur91] S. Muraki. Volumetric shape description of range data using "blobby model". *Computer Graphics*, pages 217–226, July 1991. Proceedings of SIGGRAPH '91.

[Ous94] J.K. Ousterhout. *Tcl and the Tk Toolkit*. Addison Wesley, 1994.

[Ove83] M. Overmars. *The design of dynamic data structures*. Springer Verlag, 1983. Lecture Notes in Computer Science 156.

[OW90] T. Ottmann and P. Widmayer. *Algorithmen und Datenstrukturen*. BI–Wissenschafts–Verlag, 1990. Bd. 70.

[Pri57] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Tech. Journal*, 36:1389–1401, 1957.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer Verlag, 1985.

[Rad88] J.D. Radke. On the shape of a set of points. In G.T. Toussaint, editor, *Computational Morphology*, pages 105–136. Elsevier Science Publishers B.V. (North Holland), 1988.

[Rao98] S.V. Rao. *Some studies on beta-skeletons*. PhD thesis, Dept. of Computer Science & Engineering, Indian Institute of Technology, 1998.

[RM95] D. Ruprecht and H. Müller. Spatial free form deformation with scattered data interpolation methods. *Computers and Graphics 19*, pages 63–71, 1995.

[RW97] G. Roth and E. Wibowoo. An efficient volumetric method for building closed triangular meshes from 3D image and point data. In *Graphics Interface '97*, pages 173–180, 1997.

[SB97] T. Schreiber and G. Brunnett. Approximating 3D objects from measured points. In *Proceedings of 30th ISATA*, Florence, Italy, 1997.

[Sch97] T. Schreiber. Approximation of 3D objects. In *Proceedings of the 3rd Conference on Geometric Modeling*, Dagstuhl, Germany, 1997. accepted for a supplementary issue of the journal *Computing* (Springer Verlag).

[SF97] G. Soucy and F.P. Ferrie. Surface recovery from range images using curvature and motion consistency. *Computer Vision and Image Understanding*, 65(1):1–18, 1997.

[SK95] M. Stark and M. Kohler. Video based gesture recognition for human computer interaction, 1995. Research Report No. 593, Fachbereich Informatik, Lehrstuhl VII, University of Dortmund, Germany.

[ST92] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26:185–194, July 1992. Proceedings of SIGGRAPH '92.

[SvdW53] K. Schuette and B. L. van der Waerden. Das Problem der dreizehn Kugeln. *Math. Ann.*, Bd. 125:325–334, 1953.

[Tau95] G. Taubin. A signal processing approach to fair surface design. *Computer Graphics*, pages 351–358, 1995. Proceedings of SIGGRAPH '95.

[TC98] M. Teichmann and M. Capps. Surface reconstruction with anisotropic density-scaled alpha shapes. In *Proceedings of IEEE Visualization '98*, 1998.

[TL94] G. Turk and M. Levoy. Zippered polygon meshes from range images. *Computer Graphics Proceedings, SIGGRAPH '94, Annual conference series*, pages 311–318, 1994.

[TM91] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[TWK88] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[Vel94] R.C. Veltkamp. Closed object boundaries from scattered points. In *Lecture Notes in Computer Science 885*. Springer Verlag, 1994.

[Vel95] R.C. Veltkamp. Boundaries through scattered points of unknown density. *Graphics Models and Image Processing*, 57(6):441–452, November 1995.

[VMM99] J. Vollmer, R. Mencl, and H. Müller. Improved Laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, 18(3):131–138, 1999. Proceedings of EUROGRAPHICS '99, Milano, Italy.

[Vol98] J. Vollmer. Eliminierung von lokalem und globalem Rauschen in ungeordneten Punktmengen auf dreidimensionalen Oberflächen. Master's thesis, University of Dortmund, Fachbereich Informatik, Lehrstuhl VII (Computer Graphics), 1998.

[Was78] A. J. Wasserman. The thirteen spheres problem. *Eureka*, 39:46–49, 1978.

[Wel97] F. Weller. Stability of voronoi neighborship under perturbations of the sites. In *Proceedings 9th Canadian Conference on Computational Geometry*, 1997. Kingston, Ontario, Canada, August 11–14.

[Wer94] J. Wernecke. *The Inventor Mentor*. Addison-Wesley, 1994.

[Whi73] A.T. White. *Graphs, Groups, and Surfaces*. North-Holland Publ. Company, 1973.

[Wir86] N. Wirth. *Algorithmen und Datenstrukturen mit Modula-2*. B.G. Teubner Verlag, 1986.

[WM96] F. Weller and R. Mencl. Nearest neighbour search for visualization using arbitrary triangulations. In M. Göbel, J. David, P. Slavik, and J. J. van Wijk, editors, *Virtual Environments and Scientific Visualization '96*, pages 191–200. Springer Verlag New York, 1996.

[Yao82] A.C.C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11:721–736, 1982.