University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
306 14 Pilsen
Czech Republic

Methods of surface reconstruction from scattered data

State of the art and concept of doctoral thesis

Jan Hrádek

Technical Report No. DCSE/TR-2003-02
March, 2003

Distribution: public

Technical Report No. DCSE/TR-2003-02
March 2003

Methods of surface reconstruction from scattered data

Jan Hrádek

Abstract

The surface reconstruction from cloud of points is a problem, which can be encountered in many applications, from reverse engineering through CAD to e-commerce. It has become more important in the past years, when the scanning devices have become cheap enough to be available to the wide public.

This thesis addresses a problem of triangular mesh reconstruction from cloud of points embedded in E^3 without any additional information, for instance normals. It is aimed on the summation of existing reconstruction methods and the problems related to the surface reconstruction, such as normals estimation and sampling. Also our experience with the implementation of one algorithm is described.

Because the surface reconstruction is not straightforward, a plenty of methods to solve this problem exists. The methods are based on different approaches and each approach has different problems and requirements. We decided to implement an incremental construction of surface approach, and discuss it's advantages and encountered problems.

Henceforth we would like to improve our algorithm to better estimate normals, regard the estimated curvature and process in parallel.

This work was supported by the Ministry of Education of the Czech Republic – project MSM 235200005.

Copies of this report are available on
http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address :

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
306 14 Pilsen
Czech Republic

Table of Contents

1 Introduction

The problem of surface reconstruction can be stated as follows: Let *S* be a surface of object *O*, and assume that *S* is a smooth twice-differentiable two-dimensional manifold, embedded in Euclidean three-dimensional space $\mathbb{R}^3$. Given a discrete set of points P, $\boldsymbol{p}_i \in P \subset \mathbb{R}^3, i = 1, \ldots, N$, that samples surface *S*, find a surface *S′* that approximates *S*, using the data set *P*. The reconstructed mesh S' must be topologically equivalent to the surface S of the original object. For overview of the problem see figure 1.1.

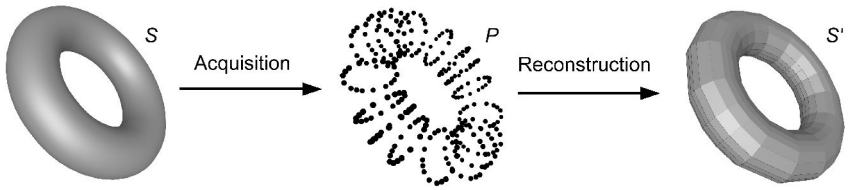


Figure 1.1: Overview of the problem of the surface reconstruction.

Such problem can be encountered in many branches, for example in computer vision, pattern recognition and reverse engineering, thus large effort has been invested in this area. As a result of this effort, and the fact that the solution of this problem is not straightforward, many methods were developed to solve this problem.

The methods of surface reconstruction can be divided into groups, according to some criteria. The most important classification of methods is done by the initial approach: some methods select the output triangle mesh from an initial triangulation, other methods create a volumetric representation from the points that is further processed, other methods construct incrementally a triangular mesh from an initial element (a point, edge or triangle) and the last sort of methods deform an initial surface along the measured points.

Some methods require additional data and the normal vectors in data points are demanded the most. Such data can be acquired during the acquisition process called the acquisition pipeline or may be estimated from data *P*.

The triangular mesh is the most demanded output of the surface reconstruction, because the key problem of the surface reconstruction is to capture the topology. The triangular mesh is simple and efficient representation of topology, but if other properties are required, a better representation is needed. For example, if C^1 continuity is demanded the output surface should be in higher order.

This thesis addresses the problem of meshing a surface only known by an unorganized set of points with a set of triangles.

The document is structured as follows. In the following part of the introduction the acquisition pipeline is described in short along with some sampling criteria. The detailed overview of some current methods of surface reconstruction is in Chapter 2. The recent implementation of one reconstruction algorithm and the results are described in Chapter 3.

1.1 The acquisition pipeline

The acquisition pipeline [25], [12] depends on the method used to scan the surface. It is possible to use mechanical probes to scan the surface through physical contact, however non-intrusive techniques are more popular, because these techniques can reliably scan soft materials such as human skin. One of these techniques is called *range scanning* – the distance from the scanning device to the sample points on the object

surface is estimated using optical triangulation, interferometric technique using laser light or "time of flight" (radar) principle.

The usual acquisition is completed in these successive steps:

- *Calibration*: The acquisition system parameters are estimated according to hardware and environmental characteristics. Calibration is prerequisite for obtaining accurate measurements. It needs to be done every time the acquisition parameters change.
- *Scanning*: The object surface is sampled from one view, resulting in a densely sampled part of the surface. It is often needed to scan the surface from many viewpoints to get a set of samples covering the complete surface.
- *Registration*: The sampled parts of the surface must be merged together. For example in range scanning techniques, the acquired range scans reside within their own local coordinate system, thus they have to be aligned with each other to express them in a global frame.

The acquisition must be done precisely, i. e. the surface must be well sampled.

1.2 Sampling

The reliability of the reconstructed surface depends on the amount of information available about the surface, whatever method is used to perform the reconstruction [7]. This amount of information is related to the term of sampling. If the surface S is not properly sampled by set of points P, then the areas where the sampling is insufficient can not be reconstructed successfully.

There are some theorems specifying the term "sufficient sampling". Their description follows:

- *Local feature size*: "A good sample is one in which the sampling density is (at least) inversely proportional to the distance to the medial axis." Specifically, a sample $\boldsymbol{p}$ is an *r-sample* from a surface S when the Euclidean distance from any point $\boldsymbol{m} \in S$ to the nearest sample point $\boldsymbol{p}$ is at most r times the distance from $\boldsymbol{m}$ to the nearest point of the medial axis of S. The constant of proportionality r is generally less than one. In [2] is observed that in practice $r = 0.5$ generally suffices.
- *Sampling path*: "The sampling path [6] $P \subset S$ is said to sample S with the sampling path ε if any sphere with radius ε and center in S contains at least one sample point in P." The sampling path theorem is formally similar to the one stated by Hoppe [23]. Hoppe stated that the set P is ρ-dense and δ-noisy that means the sampling path is ρ and the maximum displacement of the samples regarding to the original sampled point on surface, i.e. noise, is δ. The sampling path is uniform sampling where maximal distance between samples is ε. The well known Shannon sampling theorem can be easily extended to E^3: the value of ε must be lower than half the size of the smallest detail.

The second theorem is more common, because the most of the acquisition techniques yield uniform data and thus is more widely used. However note that the sampling path is a global criterion for sampling and thus the method that needs the data set of the object sampled using sampling path theorem needs more samples than the method that needs the data set sampled using local feature size theorem, and therefore the complexity of the reconstructed mesh is bigger. Examples of sampling an object using either of the theorems are in figure 1.2.

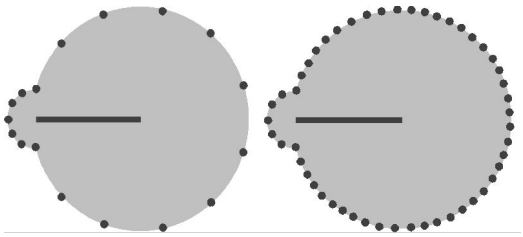


Figure 1.2: Sampling of a 2D object using the two different sampling theorems. On the left figure is the object sampled to comply the Local Feature Size theorem. On the right is the object sampled to comply the Sampling path theorem. The gray area is the sampled object itself, black dots are the samples and the black line is the medial axis of the object.

The surface is properly reconstructed:

- If the sampling density is sufficient for the reconstruction algorithm and
- the surface sampled have the necessary properties, which are required by the reconstruction algorithm.

The output of the reconstruction algorithm may have some disadvantages and therefore various post-processing steps are usually applied afterwards.

1.3 Post-processing of the reconstructed triangular mesh

The output of the reconstruction algorithms is correct but may have some bad properties such as too large number of triangles in regions of low curvature, the sampled points contained noise etc.

In the case the surface was oversampled to strengthen the robustness of the reconstruction the final triangular mesh has areas filled with a large number of triangles that may be represented with fewer triangles. Methods that handle the conversion of the mesh with large number of triangles to the mesh with fewer triangles are called *decimation* or *mesh reduction* techniques. These techniques observe the angles between neighboring triangles or curvature of the mesh and uses these information to reduce the complexity of the mesh while preserving the shape of the mesh. A detailed overview of the decimation techniques can be seen in [21].

When the sampled points are influenced by the noise and a interpolation reconstruction technique (see Chapter 2) is used, the surface of the reconstructed shape will be noisy too. Therefore such surface is often filtered to remove noise, but retain the details which are contained in the shape. This process refers to the *smoothing* of the surface. During the smoothing the surface is not reduced in anyway, but simply the data points are moved towards new locations to reduce the noise they cause. A work covering the problem of smoothing and denoising a triangular surface was made recently [33].

The output of the reconstruction algorithm can be further passed to *warping* algorithms (see Chapter 2.4). These algorithms need a initial approximation of the surface and this approximation is then deformed along the provided input data points. Therefore they can be chosen as the post-processing of another algorithm.

It was mentioned in the previous text, that a conversion to higher order surfaces is needed when the information about the surface topology in not sufficient for further processing. In this case the triangular surface is converted to higher order meshes such as: subdivision surfaces, spline surfaces etc. For information about the techniques of

remeshing the triangular mesh with higher order meshes see for example [19], [26], [9]. The output of the volumetric algorithms (see Chapter 2.2) can be also considered as a higher order approximation, because the intermediate output of these methods is a distance function often substituted by implicit function. One of such approach is the interpolation of implicit surfaces using radial basis functions [28].

2 Algorithms of surface reconstruction

The requirements for the reconstruction can be summarized like this:

- *Robustness*: The robustness if the most important property of the reconstruction method. The algorithm should be aware of missing samples i.e. undersampling, erroneous samples, borders etc.
- *Maximally automatic*: Preferably additional parameters are not needed for the method, just the input set of points.
- *Memory requirements*: Because the input data sets can be very large, assume over a million points, the platform or the reconstruction algorithm must be prepared for such large data sets. Because the amount of memory on todays machines is limited, the algorithm must be capable to handle very large data sets by parallel processing or some other way.
- *Speed*: The surface reconstruction is usually an off-line process, therefore the speed is one of the less important requirements of the surface reconstruction method.

The algorithms of surface reconstruction can be split into some, not necessarily disjunct, classes:

- *Approximation or Interpolation*: Interpolation means that the reconstructed surface will preserve the original data set, so the measured points will also belong to the reconstructed surface [7], i.e. $P \subset S', P \cap S' = P$. On the other hand the approximation means that the final reconstructed surface passes close by, rather than exactly through, the original sample points. The difference is illustrated on the figure 2.3.
- *Global or local*: The methods can be further classified as *global* or *local* according to the degree to which each point is considered to influence the reconstruction of the surface at distant locations: in global methods, all points are used to define the interpolant, and in local methods only close points are used to compute a "piece" of the surface [7].
- *By the approach*:

    1. *Sculpturing methods* – The sculpturing approaches first construct an initial triangulation. From this triangulation the reconstruction process then select a subset of triangles that represents the reconstructed surface.
    2. *Volumetric methods* – The volumetric methods create a volumetric form of object (as a distance function or volumetric data) at first. The mesh is reconstructed, by using the isosurface extraction techniques on the volumetric form.
    3. *Warping methods* - The warping methods deforms an initial approximation of the surface along the data points.
    4. *Incremental construction methods* – The incremental construction methods create the triangular mesh by growing the mesh along its border starting from an initial element (a point, edge or triangle).

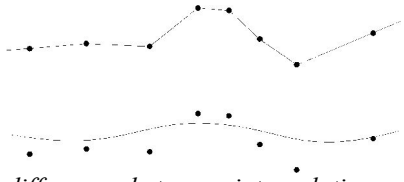


Figure 2.1: The difference between interpolation and approximation technique in 2D. On the upper part of the figure is a result of interpolation reconstruction technique and a result of some approximation technique is below it.

Each of the methods has some assumptions in order to work properly. The usual assumption is that the input data set is representing 2-manifold surface. Typically, the 2-manifold is bounded (or closed). A manifold-with-boundary is a surface locally approximated by either a disc or a half-disk. All other surfaces are non-manifold. Other very important assumption is the required sampling. Some methods require additional data along with the input data set.

In the following chapters, some of the algorithms will be described. The methods are sorted by approach. The following description omits methods that reconstruct other type of surface than the triangular mesh.

2.1 Sculpturing algorithms

The main idea of the sculpturing methods can be seen in figure 2.2. The sculpturing methods are based on selecting a set of triangles from some initial triangulation of the input points. Therefore they are global techniques. Because of reliability, robustness and theoretical guarantees the Delaunay triangulation is often taken as the initial set. The medial axis is also often considered.

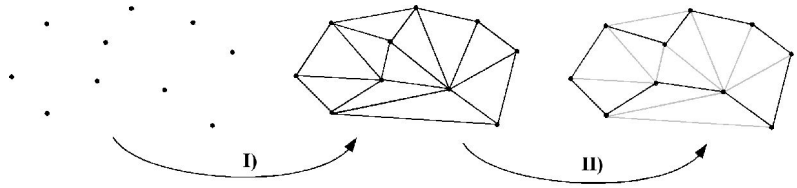


Figure 2.2: The idea of sculpturing methods (a 2D example). In the first step (I) the initial triangulation is made. Then in the second step (II) some elements from the initial triangulation are selected as the reconstructed surface.

The sculpturing algorithm can be seen as sculpturing of a spatial decomposition of space, mostly a Delaunay tetrahedronization.

The sculpturing algorithms are interpolation techniques, and their output is a triangular mesh. Their advantages are strong theoretical guarantees and the required sampling is related to local feature size. However there is also a disadvantage because the Delaunay triangulation is a lengthy process.

Delaunay triangulation, Voronoi diagram and Medial axis

Given a discrete set P of sample points in R^d, the *Voronoi cell* of a sample point is that part of R^d closer to it than any other sample. The *Voronoi diagram* is the decomposition of R^d induced by the Voronoi cells. Each Voronoi cell is a convex polytope and its vertices are the *Voronoi vertices*; when P is non-degenerate, each Voronoi vertex is equidistant from exactly $d+1$ points of P. These $d+1$ points are the vertices of the *Delaunay simplex*, dual to the Voronoi vertex. A simplex is triangle in 2D, tetrahedra in

3D etc. A Delaunay simplex, and hence each of its faces, has a circum sphere empty of other points of P. The set of Delaunay simplices form the *Delaunay triangulation*[1] of P. Thus computing the Delaunay triangulation essentially computes the Voronoi diagram as well. An example of Voronoi diagram and Delaunay triangulation of the set of points sampled from a 2D shape can be seen in figure 2.3. One of the properties of the Delaunay triangulation is, that each Delaunay simplex have a circum-circle empty of other points.

The worst-case time complexity of the Delaunay triangulation is $O(n^2)$. However as has been frequently observed, the worst case complexity for the three dimensional Delaunay triangulation almost never arises [3].

The main advantage of Delaunay triangulation is that it produces global information about the shape of the object. Although the Delaunay triangulation is a global method, the data set may be split into non-overlapping parts and processed in parallel [16].

The *medial axis* of a *(d-1)*-dimensional surface in R^d is (the closure of) the set of points with more than one closest point on the surface. The medial axis is the extension to continuous surfaces of the Voronoi diagram, in the sense that the Voronoi diagram of P can be defined as the set of points with more than one closest point in P (compare the illustrations in figure 2.3).

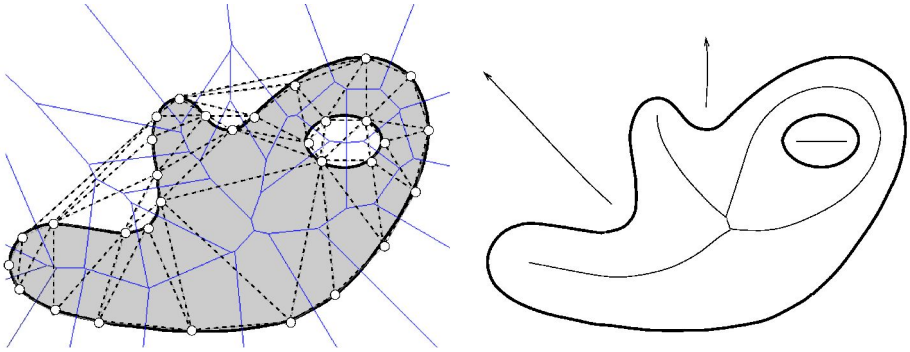


Figure 2.3: The Voronoi diagram, the Delaunay triangulation (on the left) and the Medial axis (on the right) of a 2D shape (White points - sampled vertices, Thick lines - Voronoi diagram, Dashed lines - Delaunay triangulation)

Voronoi filtering

(Two-pass) Voronoi filtering [2], [3] is based on the idea of *poles*. The poles of a sample point $\boldsymbol{p}_i$ are the two farthest vertices of its Voronoi cell, thus one on each side of the surface (see figure 2.4). Note that the pole vector approximates the normal to the surface at the given sample. Since the algorithm does not know the surface, only the sample points, it chooses the poles by first choosing the farthest Voronoi vertex regardless of direction (or a fictional pole at "infinity" in the case of unbounded Voronoi cell), an then choosing the farthest in the opposite half-space.

Denoting the poles by Q, the *crust* of P are those triangles of the Delaunay triangulation of $P \cup Q$, all of whose points are members of P. For a 2D example see figure 2.5.

1 Delaunay triangulation in 3D is often called Delaunay tetrahedronization, because as the output is a set of tetrahedra (tetrahedron is a convex hull of 4 points, ie. 4 triangular faces)

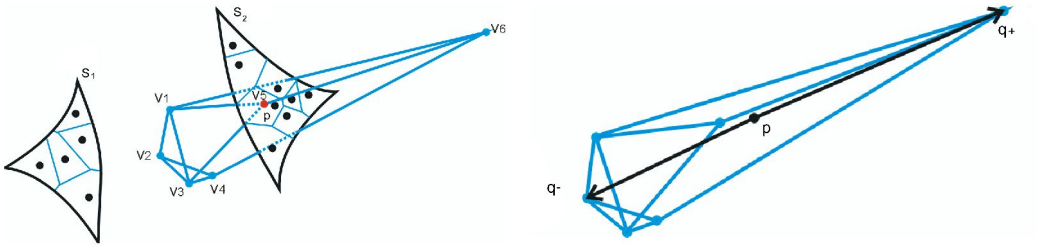


Figure 2.4: Obtaining the poles. On the left is a Voronoi cell of point P in 3D on surface S_2. The vertices of cell (V1 to V6) lie near medial axis. The surface S_1 is situated against surface S_2 and medial axis is in between them. On the right are the poles q- and q+ obtained for the point P

In 3D the crust may contain additional triangles that are not part of the surface (thin tetrahedra on surface and triangles in thin spaces between objects, see figure 2.6). Therefore two post-processing steps are used to deal with this problem.

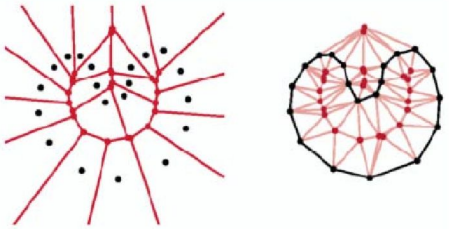


Figure 2.5: An example of crust in 2D. On the left picture is a Voronoi diagram of P. On the right picture is a Delaunay triangulation of $P \cup Q$.
(image courtesy of Nina Amenta)

The first post processing step removes triangles according to the direction of their surface normals. Let T be a triangle of the crust and let $\boldsymbol{p}_i$ be its vertex of maximum angle. This step removes T if the angle between the normal to T and the vector from any one of T's vertices to its first-chosen pole is too large. This step is called the *normal filtering*.

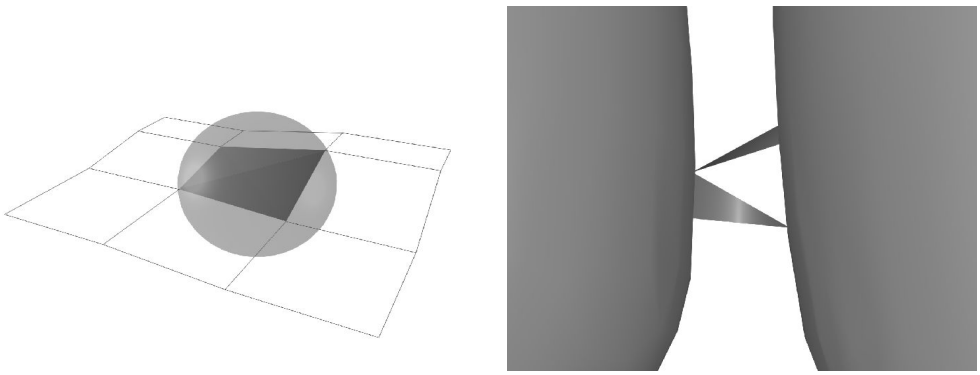


Figure 2.6: The additional triangles that were left on the surface after the CRUST sculpturing algorithm. On the left is a thin tetrahedra that was on the surface along with the tetrahedra circum-sphere and on the right are triangles connecting thin space between objects. The problem on the left is solved in the second post processing step and the problem depicted on the right is solved in the first post-processing step.

The second step removes thin tetrahedra that could be left on surface after first post-processing. In this last step, the algorithm orients all the triangles at first. It starts with a point $\boldsymbol{p}_i$ on the convex hull of P. The direction to the pole at "infinity" is called *the outside* and the direction to the second pole *the inside*. Than any triangle T incident to $\boldsymbol{p}_i$, has the outside side of T that one visible from the points on the outside direction. The

other poles of the other vertices of T are oriented to agree with this assignment. Each triangle sharing a vertex with T is oriented so that it agree on the orientations of its shared poles, and the same procedure is done by breadth-first search until all poles and triangles have been oriented.

In triangulated piecewise-linear two-dimensional manifold, two triangles meet at each edge, with outside together and inside sides together. The *sharp* edge is defined as an edge which has a dihedral angle greater than $3\pi/2$ between a successive pair of incident triangles in the cyclic order around edge, i.e. a sharp edge has all its triangles within a small wedge. The second step then greedily removes triangles with sharp edges thus it trims off pockets.

The remaining triangles form a "quilted" surface, in which each edge bounds at least two triangles, with consistent orientations. Finally it extracts the outside of this quilted surface by breadth-first search on triangles.

It is shown [2], [3] that this method is correct for well-sampled surface. The complexity of this algorithm is $O(N^2)$ where $N = |P|$, since that is the worst-case time required to compute a three dimensional Delaunay triangulation. Notice that the number of sample points plus poles is at most $3N$.

One-pass Voronoi filtering [4] is a approach similar to the two pass Voronoi filtering that has been proposed by the same authors. It computes just the Voronoi diagram of the sample points instead of the sample points and poles. The filtering is based on the following heuristics. The set of triangles T interpolating surface S must satisfy this three conditions:

I) T contains all triangles whose dual Voronoi edges intersect S,
II) each triangle in T is small, that is, the radius of its circum circle is much smaller than the distance to the medial axis at its vertices, and
III) all triangles in T are "flat", that is, the triangle normals make small angles with the surface normals at their vertices.

Assuming that S is smooth and the sampling is sufficiently dense, condition I ensures that T contains a piecewise-linear manifold homeomorphic to S. Conditions II and III ensures that *any* piecewise-linear 2-manifold extracted from T which spans all the sample points and for which every adjacent pair of triangles meets at an obtuse angle must be homeomorphic to S. In the figure 2.7 is an example of the heuristic evaluation. In this algorithm the surface normal in a sample point ($\boldsymbol{n}_1$, $\boldsymbol{n}_2$, $\boldsymbol{n}_3$ on the figure) is estimated as the vector from sample point to the farthest Voronoi vertex in Voronoi cell (a pole in Two-pass Voronoi filtering). The filtering is done by observing the angle (φ_1, φ_2, φ_3 on the figure) between the vector from sample point to intersection with Voronoi edge and the surface normal estimated in the sample point. If this angle is close to $\pi/2$ for all three Voronoi cells adjacent to the Voronoi edge (e on the figure), the dual Delaunay triangle (t on the figure) is included in the candidate set T.

Similar algorithm called *COCONE* was developed by the authors of Voronoi filtering [18]. The "*Cocone*" is the complement of a double cone (clipped within Voronoi cell Y_p) centered at sample point $\boldsymbol{p}_i$ with an opening angle $3\pi/8$ around the axis aligned with $\boldsymbol{n}$ (see figure 2.8), thus $C_p = \{\boldsymbol{y} \in Y_p : angle((\boldsymbol{y} - \boldsymbol{p}_i), \boldsymbol{n}) \geq 3\pi/8$. The triangles are selected according to this criterion: A Voronoi edge e is selected as *marked* if an intersection j of COCONE and edge e is a subset of edge e, ie. $j = C_p \cap e$, $j \subset e$. If this test succeeds for all Voronoi cells V incident to Voronoi edge e, the dual (Delaunay

triangle) of that Voronoi edge is included in the surface mesh.

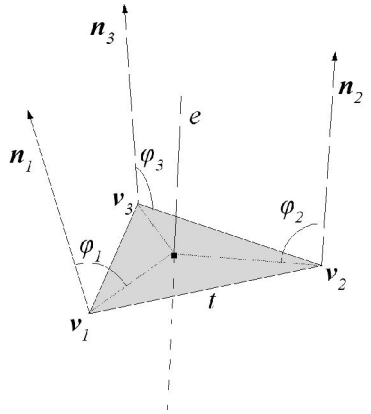


Figure 2.7: One-pass Voronoi filtering heuristic.

The computation of Delaunay tetrahedronization is computationally expensive. Therefore Dey, Giesen, Hudson [17] partitioned the input data using octree structure, applied the Cocone algorithm in each cluster, and stitched parts together. The stitch is done automatically, because each cluster in an octree box is padded with sample points from neighboring boxes that allow triangles for stitching to be computed consistently over all boxes.

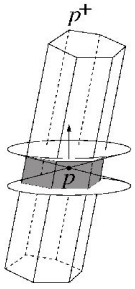


Figure 2.8: COCONE of sample point p in it's Voronoi cell (image courtesy of Tamal K. Dey)

The Power crust algorithm [5] is based on the construction of the approximation of *medial axis transformation* (MAT) that is further used to produce the piecewise-linear surface approximation. The MAT is a representation of the object as the infinite union of its maximal internal balls. As approximation of MAT the *polar balls*, a subset of the Voronoi balls (a circumsphere centered at Voronoi vertex and touching the nearest samples), are used. Polar balls are only those Voronoi balls, whose center is the pole (see Two-pass and One-pass Voronoi filtering for poles). This can be omitted in 2D and all Voronoi vertices can be selected as centers of power balls. In 3D however only poles must be selected. The polar balls belong to two sets, one more or less filling up the inside of the object, and the other the outside. The polar balls are used to construct *the power diagram* through a computation of *weighted Voronoi diagram*. The *power diagram* is a subdivision of space into polyhedral cells, each cell consisting of the points in R^3 closest to a particular power ball, under a convenient distance function, the *power distance*. The power distance between an unweighted point $\boldsymbol{x}$ in R^3 and polar ball $B_{c,r}$ (ball with center $\boldsymbol{c}$ and radius r) is

$$d_{pow}(\boldsymbol{x}, B_{\boldsymbol{c},r}) = d^2(\boldsymbol{c}, \boldsymbol{x}) - r^2 \qquad (2.1)$$

where function *d* represents the usual Euclidean distance. In figure 2.9 is a 2D example of power crust algorithm.

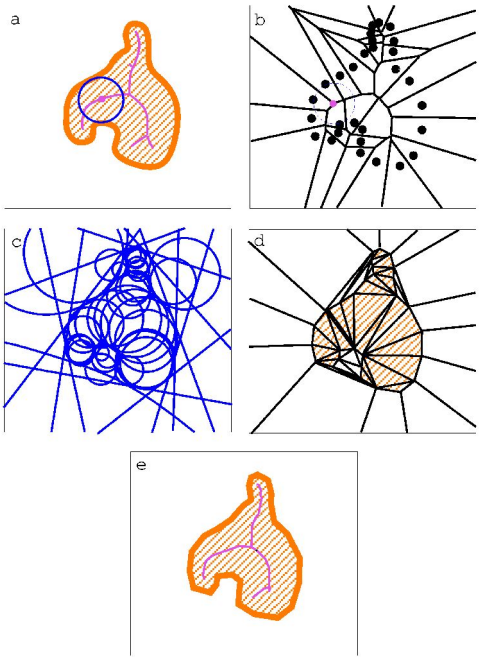


Figure 2.9: 2D example of power crust construction. a) An object with its medial axis. b) The Voronoi diagram of S. c) The inner and outer polar balls. d) the power diagram cells of the poles. e) The power crust and the powershape of its interior.
(image courtesy of Nina Amenta)

The boundary separating the cells belonging to inner polar balls from the cells belonging to outer polar balls is a piecewise-linear surface that is the output, the *power crust*.

Compared with the One and Two pass Voronoi filtering (the Crust algorithms) the Power crust algorithm does not need any clean-up or post processing steps.

Normalized meshes

The normalized mesh [6] of surface *S,* associated to sampled point set *P,* is the set of Delaunay *k*-simplices $T_j=[\boldsymbol{p}_1,\ldots,\boldsymbol{p}_k]$ with $\boldsymbol{p}_i\in P$ for which there exist a point $m\in S$ so that $d(m,\boldsymbol{p}_1)=\ldots=d(m,\boldsymbol{p}_k)=d(m,P)$. By definition, the normalized mesh is included in the Delaunay diagram of *P*. It is formed by the Delaunay elements (edges, faces and simplices) whose dual (Voronoi edge) intersects the surface. Normalized mesh provides a piecewise linear interpolant of the surface that converges to the surface when the sampling path tends to 0. In order to simplify the search for the normalized mesh, S as the boundary of an *r-regular shape* is assumed. Let $\boldsymbol{B}_0$ be the unit ball. A shape *X* is said to be *r-regular* if it is morphologically open and closed with respect to a disk of radius $r>0$:

$$X=(X\ominus rB_0)\oplus rB_0=(X\oplus rB_0)\ominus rB_0 \qquad (2.2)$$

where $\oplus$ means morphological dilatation and $\ominus$ means morphological erosion. The *r*-regular shape (see figure 2.10) has some useful properties: the boundary of *r*-regular shape has at each point a tangent and a radius of curvature greater or equal to *r*, the boundary of a *r*-regular shape divides any ball with radius *2r* and center on the boundary

in exactly two connected components. From the previous property, it follows that if the sampling path $\epsilon < 2r$, the normalized mesh provides a tilling of the surface, ie. the normalized mesh retains all the topological properties of the surface. The surface is assumed to be the boundary of an *r*-regular shape.

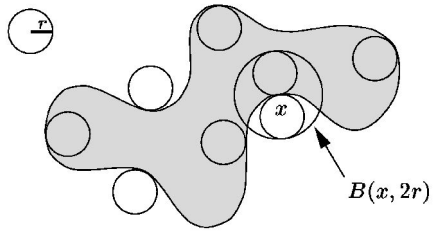


Figure 2.10: The 2D r-regular shape.
(image courtesy of Dominique Attali)

To detect the boundary face the following property is used. In 2D Delaunay circum-circles tend to maximal circum-circles of the object and to the complement of the object when the sampling tends to 0. Consequently, Delaunay circum-circles become tangent to the boundary. In 3D, the angle formed by the two Delaunay spheres passing both sides of this triangle is measured, to evaluate the belonging of a Delaunay triangle to the surface.

According to the figure 2.11: If *T* is a Delaunay triangle, c_1 and c_2 the centers of the two Delaunay spheres passing through *T* and ***p*** a vertex of *T*, the angle can be evaluated:

$$\delta(T) = \pi - \widehat{c_1 p c_2} \tag{2.3}$$

If the sampling path is $\epsilon < \sin(\pi/8) r$, then the set of triangles $S_{\pi/2}$, i.e. the set of triangles for which $\delta(T) \leq \pi/2$, is the normalized mesh of *S* associated to the points *E*. If the angle $\delta(T)$ is near 0, there is every chance that the triangle *T* belongs to the boundary. If the angle is near π, the triangle has every chance to be inside or outside the object.

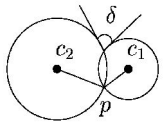


Figure 2.11: The angle formed by Delaynay circum-circles (2D case).
(image courtesy of Dominique Attali)

Two propositions are made and proved for 2D case:

- Let *X* be an *r*-regular shape and *P* be a sample of the surface of *X* with sampling path ϵ. For each edge ***(pq)*** of the Delaunay digram of *P*:
  - if ***(pq)*** belongs to the normalized mesh and $\epsilon < r/2$, then $\delta(\boldsymbol{pq}) < \pi/2$
  - if ***(pq)*** does not belong to the normalized mesh and $\epsilon < \sin(\pi/8) r$, then $\delta(\boldsymbol{pq}) > \pi/2$
- Let *X* be and *r*-regular shape and *P* be a sample of surface of *X* with sampling path ϵ. If $\epsilon < \sin(\pi/8) r$, then the set of triangles *T* for which $\delta(T) \leq \pi/2$ is the normalized mesh.

Thus the algorithm, first compute the Voronoi diagram of input set and then according to the angle of the Delaunay spheres removes triangles. In 3D case some Delaunay spheres intersect the surface without being tangent to it (see figure 2.12), thus leaving some holes after triangles removal. Note that after the triangles removal the surface doesn't contain erroneous triangles (triangles that don't belong to the surface).

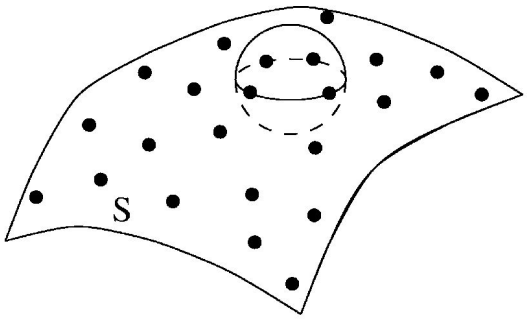


Figure 2.12: Delaunay sphere intersecting surface.
(image courtesy of Dominique Attali)

Two approaches are proposed to fill those holes:

1. Triangulate each polygon of the hole border independently from the others (done by adding one after the other Delaunay triangles sharing two or three edges with this polygon),
2. A volume tessellation, ie. merge Delaunay tetrahedra until space is partitioned into satisfactory number of objects (this approach assumes surfaces to be closed and is less general than the previous one).

Heuristic sculpturing approaches

Boissonnat [14] proposed a sculpturing heuristic for selecting a subset of Delaunay tetrahedra to represent the interior of object. From the Delaunay triangulation, 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. Tetrahedra 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 (see figure 2.17). This decision value is usefull, because flat tetrahedra of the Delaunay triangulation tend to be outside of the object. The algorithm stops if all points lie on the surface, or if the deletion of the tetrahedron with the highest decision value does not improve the sum taken over the decision values of all tetrahedra incident to the boundary of the polyhedron.

This heuristic is motivated by the observation that “typical” Delaunay tetrahedra have circum spheres approximating maximum empty balls centered at points of the medial axis.

A *hybrid approach* proposed by Attene, Spagnuolo [7] is based on Boissonnat sculpturing and on use of some interesting properties of geometric graphs. The EMST is used as a constraint during the sculpturing of the Delaunay tetrahedronization of the data set, and in addition another constraint is used, the so-called *Extended Gabriel Hypergraph* (EGH). The algorithm starts with the generation of the Delaunay

triangulation of the vertices *P*; then tetrahedra are iteratively removed from this triangulation, until all vertices lie on the boundary.

The concept of extended Gabriel hypergraph is used to locate, inside the Delaunay triangulation, those triangles that have a high probability of being close to the original surface. The Gabriel graph of *P* is the maximal graph $GG(P)=(P,E)$ defined by $E\subseteq P\times P$ and $E=\{e_k=(\boldsymbol{p}_i,\boldsymbol{p}_j)\ k=1,\ldots,n$ / the smallest sphere containing points $\boldsymbol{p}_i$ and $\boldsymbol{p}_i$ does not contain any other point of P . A 2D example of Gabriel graph is in figure 2.13.

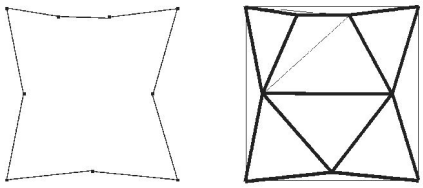


Figure 2.13: Example of the boundary of a point set in 2D and corresponding Delaunay triangulation. In the DT the Gabriel graph edges are highlighted.
(image courtesy of M. Attene)

Given *P*, and given the associated *GG(P)* = (*P*,E_{GG}), the extended Gabriel hypergraph *EGH*(*P*) is defined as *EGH*(*P*) = (*P*,E_{EGH},*T*) such that E_{EGH}, the edge set, is initially defined by E_{GG} while *T*, the triangle set, is initially empty. Both final sets are constructively defined as follows:

- $\forall e_1, e_2\in E_{GG}, e_1=(\boldsymbol{p}_1,\boldsymbol{p}_2)$ and $e_2=(\boldsymbol{p}_2,\boldsymbol{p}_3)$, if $\boldsymbol{p}_1,\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are not aligned and if the smallest sphere for $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ does not contain any other points of *P*, then $E_{EGH}=E_{EGH}\cup\{(\boldsymbol{p}_1,\boldsymbol{p}_3)$
- any cycle of three edges in E_{EGH} defines a new triangle in *T*.

If the Delaunay triangulation of the data set *P* is unique, then all the triangles of the Extended Gabriel hypergraph are triangles of the Delaunay triangulation.

The algorithm works as follows:

1. Construct Delaunay triangulation
2. Construct a heap containing *removable* tetrahedra sorted with a *criterion*
3. While there still exist vertices not lying on the boundary and the heap is not empty, get topmost tetrahedron from heap and if its removable, remove it and insert each new removable tetrahedron into the heap.

A tetrahedron *H* is removable if an only if:

1. If tetrahedron *H* has three boundary triangles, it should not be removed.
2. If tetrahedron *H* has exactly two faces on the boundary with a common edge *e*, it can be removed only if the edge opposed to *e* is not already on the boundary.
3. If tetrahedron *H* has only one face on the boundary, it can be removed only if the vertex opposed to that face is not already on the boundary.
4. If tetrahedron *H* has only one face on the boundary, that face must not belong the *EGH*.
5. If tetrahedron *H* has two faces on the boundary, these must not belong of *EGH*,

moreover the common edge of the two faces must not belong to the *EMST*.

The criterion used for sorting the tetrahedra into the heap can be the same as decision values used by Boissonnat or other. The authors also claim that the tetrahedra that have the longest edge on the boundary have to be removed first as another possible criterion.

Until this point the algorithm is able to reconstruct surfaces of genus 0. EMST is again employed to process the surfaces with higher genus. After the previous sculpturing process the following steps are performed:

1. Consider all the removable tetrahedra whose removal adds one edge of the EMST to the boundary.
2. Remove such tetrahedra.
3. If there exists an edge *e* of the EMST that is not on the boundary and it is possible to create a hole, create a hole according to *e*, and start again the previous sculpturing.

Authors call the previous part *constrained sculpturing* and this part the *not constrained sculpturing*. The first part is used to perform main sculpturing process and the second part creates holes if possible. The example of reconstruction of the torus is in figure 2.14.

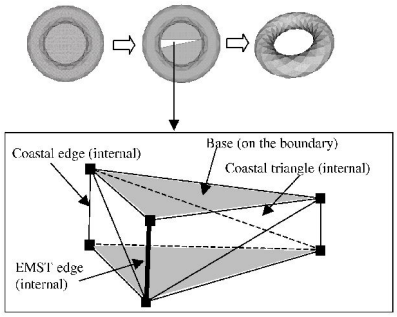


Figure 2.14: A reconstruction of torus. This figure demonstrates the purpose of the not constrained part of the algorithm.
(image courtesy of M. Attene)

Alpha shapes

Conceptually, *α-shapes* [20] are a generalization of the convex hull of a point set. Let α be a real number with $0 \leq \alpha \leq \infty$. The α-shape of *P* is a polytope that is neither necessarily convex nor necessarily connected. For $\alpha = \infty$, the α-shape is identical to the convex hull of *P*. However, as α decreases, the α-shape shrinks by gradually developing cavities. These cavities may join to form tunnels, and even holes may appear (see figure 2.15).

Intuitively, an *α-shape c*an be obtained as follows: Consider a ball-shaped eraser, of radius α, and think of *P* as a set of points in space that the eraser cannot inter-penetrate. Imagine moving the eraser everywhere in space, removing all simplices that the eraser can pass through (remember that the eraser is constrained by the data points). All that is left after the erasing constitutes the α-shape of *P*. As α varies, one can obtain different α-shapes. For example, for α=0 the α-shape is *P* itself. For $\alpha = \infty$ one obtains the convex hull of *P*.

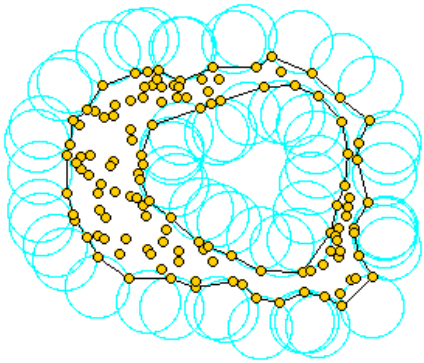


Figure 2.15: 2D example of Alpha-shapes.

For $0 \le \alpha \le \infty$, let an α-*ball* be an open ball with radius α. An α-ball b is *empty* if $b \cap P = 0$. For $0 \le k \le 2$, a k-simplex σ_T is said to be α-*exposed* if there is an empty α-ball b with $T = \delta b \cap P$, where δb is the sphere or plane bounding b. A fixed α thus defines sets $F_{k,\alpha}$ of α-exposed k-simplices for $0 \le k \le 2$. The α-shape of P, denoted by A_α, is the polytope whose boundary consists of the triangles in $F_{2,\alpha}$, the edges in $F_{1,\alpha}$ and the vertices in $F_{0,\alpha}$. For more details see [20].

The α-shapes are closely related to Delaunay triangulation. By definition, for each simplex σ_T of Delaunay triangulation, there exist values of $\alpha \ge 0$ so that σ_T is α-exposed. Conversely, every face of A_α is a simplex of Delaunay triangulation. Therefore the relation of the α-shapes with the Delaunay triangulation is that the Delaunay triangulation is a union of all α-exposed k-simplices, for $0 \le \alpha \le \infty$. Thus the Delaunay triangulation can be used to represent the family of α-shapes of P.

α-shape is thus a sub complex of Delaunay triangulation, but problem is the selection of optimal α that yield the α-shape whose boundary represents the reconstructed surface. There exists a commercial package from Raindrop Geomagic based on Alpha shapes. An alpha-shape algorithm with cleanup-phase has been proposed recently [10].

α-*solids* [8] is a combination of α-shapes and Boissonat sculpturing approach [14]. This technique has two phases:

In the first phase an approach similar to α-shapes is used to select a subset of the Delaunay triangulation of P enriched with some additional points that lie far from the data set P and that form an envelope of the data set: Let a (real positive) value for the parameter α be given. Then do breadth-first search of the adjacency graph of DT (consider only adjacencies among tetrahedra), starting from tetrahedra that are *outside* the shape, i.e. one or more of its faces are part of the border of the convex hull of the Delaunay triangulation, and mark those tetrahedra as *external,* see figure 2.16. However do no cross the common face between two adjacent tetrahedra if all three edges of the face have length that is less than α. All tetrahedra that are still unmarked after the traversal are then marked as *internal*. Their union will be called the α-*solid*. The difference of between this part of algorithm and α-shapes is that the eraser cannot go inside the shape if it cannot "pass through" its boundary.

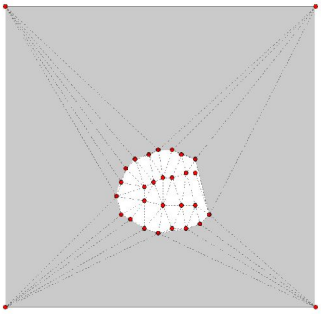


Figure 2.16: Tetrahedra "outside" the shape (2D case). On the figure is an example of Delaunay triangulation of 2D point set enriched with additional points (in the corners of the figure). In this Delaunay triangulation are boundary triangles (tetrahedra in 3D) treated as outside, i.e. external (gray triangles).

The smallest value of parameter α needs to be chosen, such that the solid selected by the first part has the following characteristics:

1. It is connected.
2. All the data points are on its boundary or in its interior.
3. Its boundary is a two-manifold.

The value of the parameter is selected by binary search over the length of all edges, taking α as the value of selected edge length.

In the second phase approach similar to local Boissonat sculpturing strategy is taken, but based on different geometric criterion. All tetrahedra, what have one or more boundary faces, are inserted into a priority queue, where the maximal priority is given to the tetrahedra with the largest value of the maximal distance between boundary face and their circumscribing sphere, see figure 2.17. These candidate tetrahedra are then extracted from the queue one by one and considered for removal. The candidate tetrahedron is removed if and only if:

1. It has one boundary face and the opposite vertex is internal.
2. It has two boundary faces, the opposite edge does not lie on the boundary, and the *local smoothness criterion* is satisfied (see below).

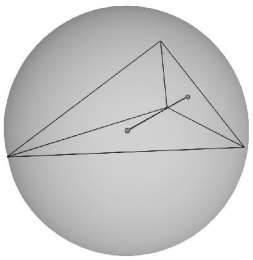


Figure 2.17: Priority measure. On the figure is a tetrahedron, its circumsphere and the line between the dots represent the priority measure, i.e. the maximal distance between the center of a face of the tetrahedron and the center of the circumsphere of the tetrahedron.

When only these two types of tetrahedra are removed, the boundary of the remaining set of tetrahedra is guaranteed to remain a manifold [8].

The local smoothness criterion used above is stated as follows: Consider a tetrahedron having exactly two boundary faces. These faces form a dihedral angle γ_0. They also form four dihedral angles $\gamma_1 ... \gamma_4$ with adjacent boundary faces, see figure 2.18. Let Γ be the following sum:

$$\Gamma = \sum_{i=0..4} |\pi - \gamma_i| \qquad (2.4)$$

Γ gives a measure of the "local smoothness" of the mesh: If the dihedral angles formed by all adjacent boundary faces are all close to straight angles, then Γ is small. Assuming that the tetrahedron is removed, its two internal faces become boundary faces. It can be measured what the new local smoothness, say Λ, would be, and Γ compare with Λ. The local smoothness criterion is therefore the following: remove the tetrahedron if and only if $\Gamma > \Lambda$, that it, if the local smoothness improves.

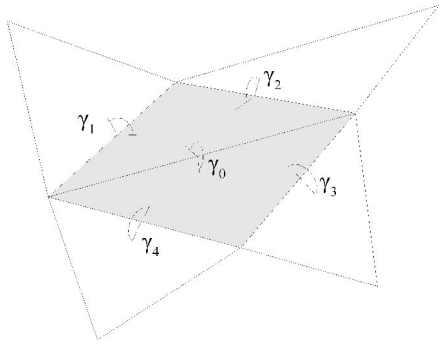


Figure 2.18: Smoothness criterion. The two boundary faces of a tetrahedron are gray. They form a dihedral angle γ_0, and dihedral angles $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ with neighboring boundary faces.

The algorithm might get stuck in a local minimum. For example, at some point it might become impossible to remove any tetrahedron, because all have two boundary faces and none satisfies the smoothness criterion. A "look-ahead" search is therefore done in such case to remove a tetrahedron that does not satisfy the criterion: It is removed, if some other tetrahedron adjacent to it consequently becomes a candidate, thus the smoothness criterion is satisfied. The depth of such look-ahead search can be limited, for all practical purposes, to about 10 removed tetrahedra.

The alpha solid technique has been used by the same authors [8] for an estimation of the so called "distance function". Distance functions is the key concept of the following sort of algorithms.

2.2 Volumetric algorithms

Volumetric algorithms are based on the idea of turning point cloud representation of surface to volumetric representation of objects. General idea can be seen in figure 2.19. Volumetric representation may be a 3D grid (uniform or nonuniform such as octrees) of values, that describe some function in space in this case a *distance function* or some sort of mathematical description of this distance function.

Figure 2.19: Overview of a volumetric algorithm.

A distance function can be defined as follows: A function $f : D \to R$, where $D \subset R^3$ is a region near the data, such that f estimates the signed geometric distance to the unknown surface S [23]. In other words, the distance function is a implicit function describing the approximation of the surface M.

The surface of the object is extracted from volumetric representation using the methods for isosurface extraction, usually with some sort of *Marching cubes algorithm* [27]. Therefore these methods are approximation techniques and the output is often higher order mesh.

The distance function is closely related to the normal vectors of the surface. The knowledge of the normal vectors along with the data point helps the obtaining of the distance function.

For example the algorithm of Pulli et al. [29] uses range maps instead of point cloud because the range maps have associated scanner position ie. the location of viewpoint. Their approach is based on a simple idea: Cut the volume as seen from the viewpoint. In this specific case the normals were not provided along with the mesh, however additional information, in this case range map along with the position of the scanner, greatly helped the reconstruction process.

Algorithm of Hoppe et al.

The algorithm of Hugues Hoppe [23] uses an tangent plane approximation of the surface to define the distance function. For each data point an oriented tangent plane is estimated as local linear approximation to the surface. For estimation of the tangent plane at given point a least squares best fitting plane in k-neighborhood of given point is used.

Let $\boldsymbol{p}_i$ be a vertex in which the tangent plane is to be approximated, and $NBHD(\boldsymbol{p}_i)$ its k-neighborhood. The tangent plane associated with point $\boldsymbol{p}_i$ is represented as a point $\boldsymbol{o}_i$ (the centroid of $NBHD(\boldsymbol{p}_i)$) and normal vector $\boldsymbol{n}_i$ at this point $\boldsymbol{o}_i$, see figure 2.20.

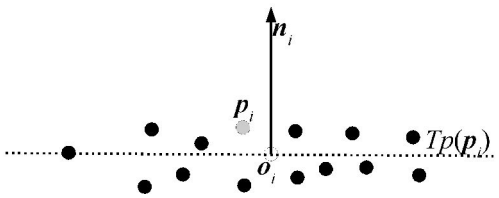


Figure 2.20: The tangent plane estimation (2D example).

To compute the normal vector $\boldsymbol{n}_i$ the covariance matrix

$$\boldsymbol{CV} = \sum_{\boldsymbol{y} \in NBHD(\boldsymbol{p}_i)} (\boldsymbol{y} - \boldsymbol{o}_i) \otimes (\boldsymbol{y} - \boldsymbol{o}_i) \quad [2] \qquad (2.5)$$

2 Where $\otimes$ denotes the outer product vector operator, thus if $\boldsymbol{a}$ and $\boldsymbol{b}$ have components a_i and b_j

of $NBHD(\boldsymbol{p}_i)$ is formed. The normal $\boldsymbol{n}_i$ is then the eigen-vector of this covariance matrix $\boldsymbol{CV}$ associated with the least eigen-value.

Notice that the tangent planes do not define the surface, they are used to define the distance function. Also note that the normal vector of the tangent plane is not oriented.

To determine the distance function, the normals of neighboring tangent planes must be oriented. The orientation is done using simple idea that geometrically close tangent planes, should be consistently oriented, therefore the consistent tangent plane orientation can be modeled as graph optimization.

The graph contains one node N_i per tangent plane $Tp(\boldsymbol{p}_i)$ with and edge *(i,j)* between N_i , N_j if the tangent plane centers are sufficiently close. The cost on edge *(i,j)* encodes the degree to which N_i , N_j are consistently oriented and is taken to be $\boldsymbol{n}_i \cdot \boldsymbol{n}_j$ (dot product). The problem is to select orientations for the tangent planes so as to maximize the total cost of the graph. However this is computationally very expensive, thus approximation is used. As graph the euclidean minimum spanning tree (EMST) enriched by some edges (the edge *i, j* is added if either $\boldsymbol{o}_i$ is in the k-neighborhood of $\boldsymbol{o}_j$ or $\boldsymbol{o}_j$ is in the *k*-neighborhood of $\boldsymbol{o}_i$) is used. This enriched graph is called the *Riemannian Graph* and encodes geometric proximity of the tangent plane centers. Normal orientation is then propagated through this Reimannian graph. The propagation favors nearly parallel planes: each edge cost function is $1 - \|\boldsymbol{n}_i\, \boldsymbol{n}_j\|$ and therefore propagation order is achieved by traversing this minimal spanning tree. Propagation done this way tend to propagate along directions of low curvature.

As the initial plane is selected the one, for which the centroid $\boldsymbol{o}_i$ has the largest *z* coordinate. Such plane must have positive *z* coordinate of normal vector. The propagation is done by depth-first traversing of the MST.

After the normal orientation the distance function can be determined correctly. In short the distance function can be described as follows: for a given point $\boldsymbol{x}$, a distance function is the distance to the nearest tangent plane (nearest in the sense point $\boldsymbol{x}$ to center $\boldsymbol{o}_i$), see figure 2.21. The sign is determined according to the normal of the nearest tangent plane.

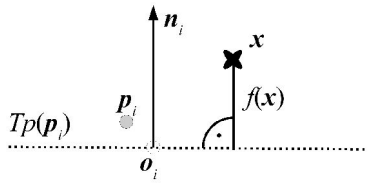


Figure 2.21: Determining the distance function for point $\boldsymbol{x}$.

The distance function $f(\boldsymbol{x})$ is defined as: Given a point $\boldsymbol{x}$, at first find a tangent plane $Tp(\boldsymbol{p}_i)$ whose center $\boldsymbol{o}_i$ is closest to $\boldsymbol{x}$. This tangent plane is a local linear approximation to *S*, so the signed distance $f(\boldsymbol{x})$ to surface is taken to be the signed distance between $\boldsymbol{x}$ and its projection *z* onto $Tp(\boldsymbol{p}_i)$, thus $f(\boldsymbol{x}) = dist_i(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{o}_i)\, \boldsymbol{n}_i$

.

This rule must be extended to handle surfaces with boundaries. Assume, that the data point are from a set that is ρ-dense and δ-noisy, thus the sample points don't leave holes of radius larger than $\rho + \delta$. Therefore the algorithm marks all points $\boldsymbol{x}$, for which the euclidean distance between its projection *z* to tangent plane and the point $\boldsymbol{p}_i$ from data

---

respectively, then the matrix $\boldsymbol{a} \otimes \boldsymbol{b}$ has $a_i b_j$ as its ij-th entry.

set nearest to z: $d(\boldsymbol{p}_i, z) < \rho + \delta$, as *undefined*.

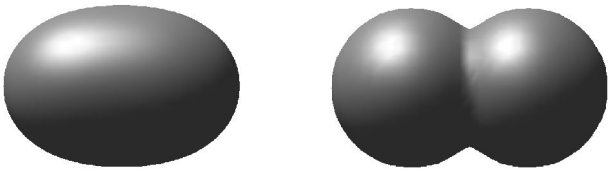
The distance function is represented by using uniform grid – a volume, thus it is discretely sampled over finite domain. The size of the cell should be less than $\rho + \delta$. The volume is processed using a contour tracing algorithm (like Marching cubes), only where the distance function is low and defined, thus giving rise to boundaries in the simplicial surface.

The final triangular mesh is further improved [22].

Medial axis combined with distance function

The algorithm of Bittar, Tsingos and Gascuel [13] is the combination of medial axis and distance function. This algorithm has two phases: determination of medial axis and implicit surface reconstruction.

1. *The determination of medial axis.* The input point set is embedded in uniformly subdivided volume. Each voxel (a cell of volume) containing one or more points is said to lie on boundary thus its distance from the boundary is 0. Starting from the voxels that lie on boundary the distance is recursively determined to other voxels inside the volume, by breadth-first search through volume. Voxels with local maximal value form the medial axis of the object.
2. *The distance function.* The distance function is in this case an implicit function defined as an union of balls centered at medial axis voxels and a radii of values at these voxels. Each ball is defined as a "*field function*". This function describes the behavior of the union of balls. If the function is "soft" (ie. blends slowly over the definition interval) the union looses sharp edges and details, however if the function is "sharp" (ie. it changes fast over the definition interval) the details and sharp edges are preserved, see figure 2.22. The surface can then be extracted using methods for isosurface extraction.



(a) "soft" field function (b) "sharp" field function

Figure 2.22: The effect of field function on the reconstructed surface. (image courtesy of E. Bittar)

The key problem of this method is the determination of the grid size. Small resolution will lead to detail loss, very large resolution may break the surface, see figure 2.23.

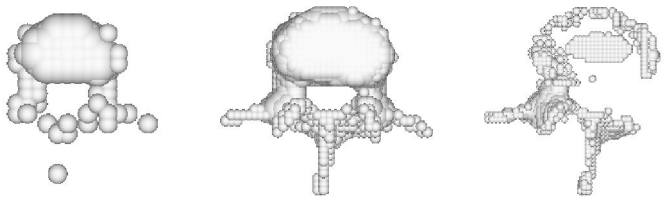


Figure 2.23: The impact of the resolution of the grid on the reconstruction. On the left the grid is too small to capture all the details and on the right the grid is too big and therefore the surface is broken.
(image courtesy of E. Bittar)

Approach of Roth & Wibowoo

The approach of Roth and Wibowoo [30] is based on estimating the normals at each point like the approach of Hoppe.

The normals are estimated using the the voxel grid: For each vertex from input set $\boldsymbol{p}_i$ find two closest neighboring points $\boldsymbol{p}_i$, $\boldsymbol{p}_i$ in the voxel grid, and then compute the normal using the two points along with the original point $\boldsymbol{n}_i=(\boldsymbol{p}_j-\boldsymbol{p}_i)*(\boldsymbol{p}_k-\boldsymbol{p}_i)/\|(\boldsymbol{p}_j-\boldsymbol{p}_i)*(\boldsymbol{p}_k-\boldsymbol{p}_i)\|$.

However the normal orientation must be determined too. The normal propagation approach is used here as well. The voxels visible from the 6 axis direction must have their normals towards the viewing direction. The normal direction is fixed for those voxels, then the normal direction is propagated to neighboring voxels, see figure 2.24. This heuristic only works for closed object.

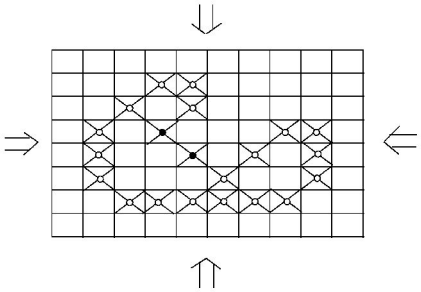


Figure 2.24: The normal orientation (2D example). The normals in voxels marked by unfilled circles are set from the orthogonal view directions, while the normals in voxels marked by filled circles are set by normal propagation.
(image courtesy of Gerhard Roth)

The normals are then smoothed using relaxation algorithm: At first a voxel whose normal most agrees with its occupied neighbors is found. This voxel is then used as the seed of a recursive smoothing algorithm. This algorithm sets each voxel normal to the average of the normals of the voxel neighbors.

For each voxel vertex the distance function is determined. The signed distance is computed by using the data points along with the estimated normals, by taking weighted average of the signed distances of every point in the neighboring cells. Then the Marching cubes algorithm is used to extract the surface.

Voronoi diagram combined with implicit functions

The approach of Boissonnat and Cazals [15] is a combination of Voronoi diagram and implicit functions.

This approach uses the so called *natural neighbors*. The natural neighbors of a point

$\boldsymbol{x}$ are defined as the neighbors of $\boldsymbol{x}$ in the Delaunay triangulation of $P \cup \{\boldsymbol{x}$. Equivalently, the natural neighbors are the points of P whose Voronoi cells are chopped of upon insertion of $\boldsymbol{x}$. More precisely, let $C_{\boldsymbol{p}i}$ be the Voronoi cell of $\boldsymbol{p}_i$ in the Voronoi diagram of P and let $V_{\boldsymbol{x}}$ be the Voronoi cell of $\boldsymbol{x}$ in the Voronoi diagram of $P \cup \{\boldsymbol{x}$. The natural region $NR_{\boldsymbol{x},\boldsymbol{p}i}$ is the portion of $C_{\boldsymbol{p}i}$ stolen away by $\boldsymbol{x}$, i.e. $C_{\boldsymbol{x}} \cap C_{\boldsymbol{p}i}$, see figure 2.25.

Let $w_{\boldsymbol{p}i}(\boldsymbol{x})$ denote the Lebesgue measure[3] of $NR_{\boldsymbol{x},\boldsymbol{p}i}$. The natural coordinate associated to $\boldsymbol{p}_i$ is defined by

$$\lambda_{\boldsymbol{p}i}(\boldsymbol{x}) = \frac{w_{\boldsymbol{p}i}(\boldsymbol{x})}{\sum_i w_{\boldsymbol{p}i}(\boldsymbol{x})} \tag{2.6}$$

This function is a continuous function of $\boldsymbol{x}$, and is continuously differentiable except the data sites. The $\lambda_{\boldsymbol{p}i}$ satisfy an identity called the *local coordinate property* (LCP), stating that $\boldsymbol{x}$ is a convex combination of its neighbors: $\sum_i \lambda_{\boldsymbol{p}i} \boldsymbol{p}_i = x$.

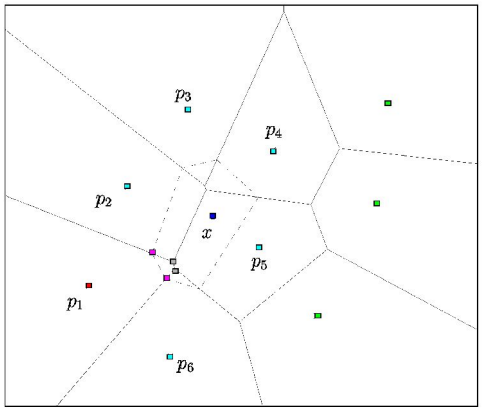


Figure 2.25: An example of natural region (in 2D). The natural neighbors of point $\boldsymbol{x}$ are points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3$, $\boldsymbol{p}_4$, $\boldsymbol{p}_5$, $\boldsymbol{p}_6$. The vertices of natural region $NR_{x, p1}$ are depicted in gray and black.
(image courtesy of J.D. Boissonnat)

When $\boldsymbol{x}$ lies outside the convex hull of P, $w_{\boldsymbol{p}i}(\boldsymbol{x})$ is unbounded if $\boldsymbol{p}_i$ is a vertex of the convex hull. In order to keep the $w_{\boldsymbol{p}i}(\boldsymbol{x})$ bounded, bounding of the domain, where the natural coordinates have to be computed, is needed. This can be easily done by adding points on a sufficiently large bounding box.

This natural neighborhood is interpolated as follows. Assume that each $\boldsymbol{p}_i$ is attached a continuously differentiable function $\boldsymbol{h}_{\boldsymbol{p}i}$ from $\mathbb{R}^d \to \mathbb{R}$ satisfying $h_{\boldsymbol{p}i}(\boldsymbol{p}_i) = 0$. The natural neighbor interpolation of the $\boldsymbol{h}_{\boldsymbol{p}i}$ is defined as $h(\boldsymbol{x}) = \sum_i \lambda_{\boldsymbol{p}i}^{1+\omega}(x) h_{\boldsymbol{p}i}(\boldsymbol{x})$, for some arbitrarily small $\omega > 0$.

The set of data points consist of the $\boldsymbol{p}_i$ plus some points $\boldsymbol{q}_i$ added on boundary box B. $\boldsymbol{h}_{\boldsymbol{p}i} = 0$ for all points $\boldsymbol{q}_i$ on the bounding box. The set P, will denote the union of the sample points $\boldsymbol{p}_i$ and of the $\boldsymbol{q}_i$, from now on. Since the $\lambda_{\boldsymbol{p}i}$ and the $\boldsymbol{h}_{\boldsymbol{p}i}$ are continuous over $\mathbb{R}^d$, h is continuous over $\mathbb{R}^d$. Moreover, $h(\boldsymbol{p}_i) = h_{\boldsymbol{p}i}(\boldsymbol{p}_i)$ since $\lambda_{\boldsymbol{p}j}(\boldsymbol{p}_i) = 0$ if $j \neq i$ and $\lambda_{\boldsymbol{p}i}(\boldsymbol{p}_i) = 1$.

The function $h_{\boldsymbol{p}_i}(\boldsymbol{x})$ is adopted from Hoppe et al. [22] as $h_{\boldsymbol{p}i}(\boldsymbol{x}) = (\boldsymbol{p}_i - \boldsymbol{x})\, \boldsymbol{n}_{\boldsymbol{p}i}$,

3 area in dimension 2, volume in dimension three.

where $\boldsymbol{n}_{pi}$ is the unit normal to S at $\boldsymbol{p}_i$. In other words, $h_{\boldsymbol{p}i}^{-1}(0)$ is the hyperplane tangent to S at $\boldsymbol{p}_i$. The final approximating surface S' is the zero set $h^{-1}(0)$.

The surface is extracted using the Delaunay triangulation, because it was already used to evaluate the natural neighbor. Define a Voronoi edge as *bipolar* if the implicit function h evaluates to a positive value at an endpoint and to a negative value at it's party. As initial approximation of the surface is selected those triangles from DT, whose dual Voronoi edge is bipolar.

2.3 Algorithms that reconstructs the surface incrementally

The algorithms that reconstruct the surface incrementally build the surface from an initial seed: point, edge or triangle. They are working locally on a small part data and thus are well suited for parallel processing of huge data sets. They build triangular meshes rather than any higher order surfaces, because of their nature.

These techniques usually require uniform sampling of surface to work properly. Because no initial structure of the data set is provided (such as Delaunay triangulation like in sculpturing methods) the vertex lookup problem must be solved along with the normal estimation in the data points.

While having these difficulties they provide linear time complexity.

Ball-pivoting approach

The approach of Bernardini, Mittleman and Rushmeier [11] imitates the ball eraser used in α-shapes.

Assume that the sampled data set P is dense enough that a ρ-ball, a ball of radius ρ, can not pass through the surface without touching sample points.

The algorithm starts by placing ρ-ball in contact with three sample points. Keeping contact with two of these initial points, the ball is "pivoting" until it touches another point. The pivoting operation is depicted in figure 2.26.

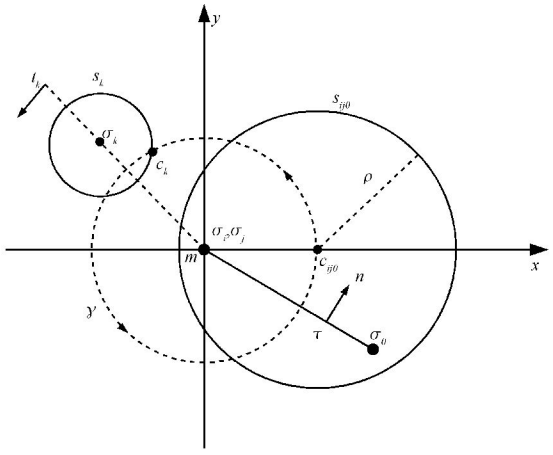


Figure 2.26: The ball pivoting operation. The ball of radius ρ *"sitting" on the triangle* τ *formed from the vertices* $\sigma_i,\sigma_j,\sigma_0$ *is pivoting around the edge* m *here perpendicular to the image. The center of the ball* c_{ij0} *rotates around the edge* m *and describes a circular trajectory* γ*. The radius of the trajectory is* $\|c_{ij0}-m\|$*. While pivoting the ball hits the point* σ_k*. The circle* s_k *is a intersection of* ρ*-ball centered at* σ_k *with* z = 0*. The circle* s_{ij0} *is the intersection of the pivoting ball with* z = 0*.*
(image courtesy of Fausto Bernardini)

The ρ-ball is pivoted around each edge of the current mesh boundary. Triplets of points that the ball contacts form new triangles. The set of triangles formed while the ρ-ball "walks" on the surface form the interpolating mesh.

The ball pivoting algorithm (BPA) is closely related to α-shapes. In fact, every triangle τ computed by the ρ-ball walk obviously has an empty smallest open ball b_τ whose radius is less than ρ. Thus, the BPA computes a subset of the 2-faces of the ρ-shape of P. These faces are also a subset of the 2-skeleton of the three-dimensional Delaunay triangulation of the point set. The surface reconstructed by the BPA retains some of the qualities of alpha-shapes: It has provable reconstruction guarantees under certain sampling assumptions and an intuitively simple geometric meaning.

The input data points are augmented with approximate surface normals computed from the range maps. The surface normals are used to disambiguate cases that occur when dealing with missing of noisy data.

Areas of density higher than ρ present no problem to the algorithm, but missing points create holes that cannot be filled by the pivoting ball. Any post-process hole filling algorithm could be employed; in particular, BPA can be applied multiple times with increasing ball radii. To handle possible ambiguities, the normals are used as stated above. When pivoting around a boundary edge, the ball can touch an unused point lying close to the surface. Therefore a triangle is rejected if the dot product of the triangle normal with the surface normal (in the vertex) is negative, see figure 2.27.

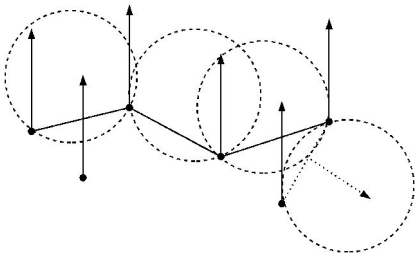


Figure 2.27: The normal check (a 2D case). The rightmost point is boundary and the ball pivots around its edge and touches a hidden point, but the dot product of the triangle normal with the vertex normal is negative, so the triangle is rejected. (image courtesy of Fausto Bernardini)

This algorithm is very efficient in time, it has linear time performance and storage requirements. However its disadvantages are the required normal vectors estimated from range maps and the selection of the radius ρ of the ρ-ball.

Approach of Huang and Menq

The algorithm of Huang and Menq [24] selects for each edge on the boundary a point, which will form a new triangle with that edge. The selection is done in tangent plane, thus the normals estimation is essential for this algorithm.

Starting from an initial triangle the mesh grows along its boundary by employing some topological operations to construct the mesh. In each successive step a triangle is added to the mesh along with some new boundary edges. For each edge on the border a so called "best point" is found or the edge is marked as final. The "best point" is then used to form new triangle.

As the initial triangle is selected the one that is formed from a vertex with maximum z-coordinate and its two closest neighbors. Normal of this triangle has positive z-coordinate, so the triangle can be easily oriented. The successive triangle's normals are oriented to coincide with the precedent triangle's normal. The edges of the initial

triangle form first border. The border of the mesh is a connected list of oriented edges.

For each edge on the border the “best point” needs to be selected. The “best point” must meet these criteria:

a) The “best point” of the edge must lie within the *k*-neighborhood of both the endpoints of the edge. The *k* is the only parameter of the reconstruction. The authors suggest $k \geq 10$.
b) For each endpoint of the edge, the “best point” must lie withing the angle formed by neighboring edges in local tangent plane, see figure 2.28. The local tangent plane is estimated using principal component analysis of the vertex *k*-neighborhood.
c) The “best point” must have minimal sum of distances to endpoints of the edge.

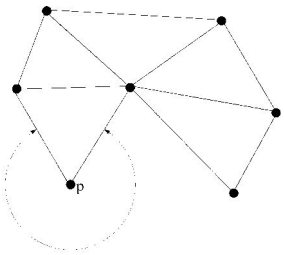


Figure 2.28: The "best point" angle in the local tangent plane. The point p lies on the boundary of the mesh. The "best points" for its adjacent edges must be in the vicinity of the angle formed by those adjacent edges. The angle is evaluated in the local tangent plane of p.

When no best point can be found, i.e. none of the points complies with the criteria, the edge is marked as *final* and it is not processed further.

Depending on the location of the best point, one of the four topological operations is used to construct the triangle, see figure 2.29. The four topological operations are:

a) *vertex joining* – The “best point” is not a part of the mesh (it is not on the border or inside the mesh). Therefore to create the new triangle, two edges are created on the border. Notice that this operation adds a point to the mesh. See figure 2.29a.
b) *face pasting* – The “best point” is already part of the surface, it is on the same border and it is a common neighbor of the edge. No new point is added to the mesh and the triangle is formed by adding one edge to the border. Notice that only this operation may remove points from the border. See figure 2.29b.
c) *ear attaching* – The “best point” is already part of the surface, is on the same border and is not a direct neighbor of the edge. Thus by adding a triangle, the border is divided into two borders that meet at the best point. The triangle is formed by adding two edges. See figure 2.29c.
d) *bridge linking* – The “best point” is already part of the surface and is not on the same border. By adding a triangle the borders will join, thus forming one border. The triangle is again formed by adding two edges. See figure 2.29d.

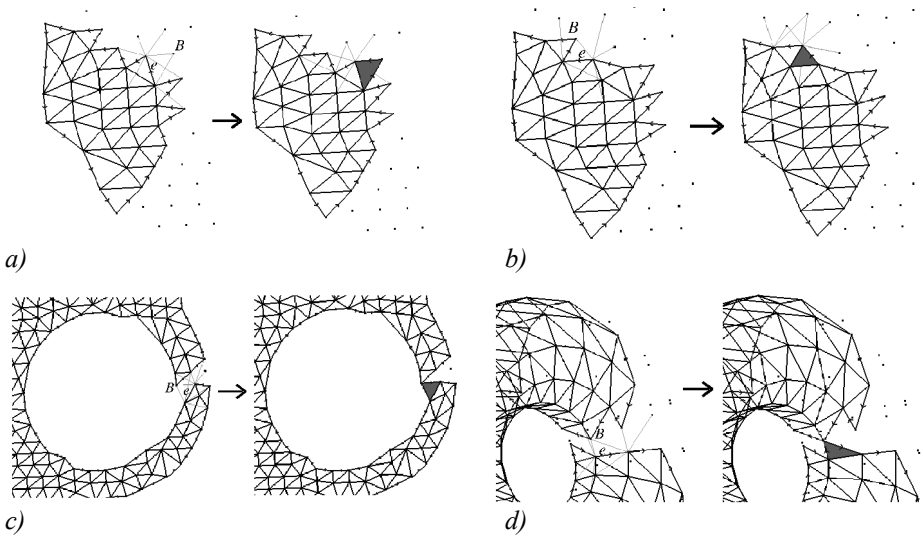


Figure 2.29: The four topological operations. On the top left (a) is the vertex joining *operation. On the top right (b) is the* face pasting *operation. On the bottom left (c) is the* ear attaching *operation. On the bottom right (d) is the* bridge linking *operation. (image courtesy of J. Huang)*

Operations vertex joining and face pasting alone are capable to reconstruct only zero genus surfaces (i.e. surfaces without holes through the object). To be able to reconstruct surfaces of any genus, the other two operations (ear attaching and bridge linking) need to be used along with the previous operations.

To improve the properties of the resulting mesh an optimization phase based on curvature estimation is used.

The advantages of the algorithm are: it can reliably reconstruct surfaces with slowly changing sampling density, it is very efficient, i.e. it has low memory consumption and linear time complexity. The disadvantage is that the tangent planes, i.e. the normal vectors in the points, needs to be estimated. Also when the sampling density changes too quickly the algorithm will fail.

The parameter K influences the resulting mesh as follows:

- If the parameter K is too low, more edges will be marked as final and therefore more holes will appear in the resulting mesh.
- If the parameter K is too high, the normals estimated will be smoothed and some details may disappear in the resulting surface.

Therefore the choice of the parameter K depends on sampling. If the sampling is nearly uniform and correct, lower K may be used. As the sampling change more over the mesh, the parameter K should increase.

2.4 Warping methods

The warping methods deform an initial surface along the sampled points. The initial surface, and therefore also the output surface, is a higher order mesh. the warping methods are approximation techniques, because the intermediate surface is deformed during the reconstruction process.

To work properly, the initial surface must be as close to the data points as possible.

Definitely, the initial surface must capture the topology of the original surface. It is possible to change the topology during the reconstruction, however these changes require estimation of additional information.

Some warping approaches are very unusual. For example one method deforms the space embeding the object to be deformed [31]. Other approach uses particle systems to model the surface [32].

Because the initial surface must be supplied the warping methods are often used as a post-processing of some other algorithm.

Approach of Algorri and Schmitt

The approach of Algorri and Schmitt [1] is based on deformation of an initial triangular mesh along the input data point using the spring model.

First and initial triangulation T' is made. This rough model of the surface is created using the partitioning of the space where points P are contained into cubes. Those cubes that contain at least one point are marked and the exterior faces not shared by any two cubes form the initial surface T'. This initial surface is further processed before the warping is used. Each vertex of the initial surface is smoothed using low pass filter (a weighted average of it's old position and the position of its neighbors) at first. Because the cubes formed a thick wall and the exterior faces lie on both sides of this thick wall an extraction of one of the sides is used afterward.

The triangular mesh is then deformed using mass-spring adaptive mesh model, i.e. the triangular mesh is deformed under the influence of attracting forces coming from the 3D points in P. Each triangle vertex in T' is a nodal mass, and each triangle edge an adjustable spring. Each nodal mass is attached to its closest point in P by an imaginary spring, see figure 2.30. The correspondence is recalculated at every iteration.

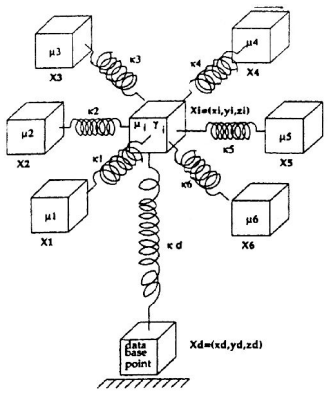


Figure 2.30: Mass-spring model. Each nodal mass is depicted as a cube. Nodal masses X1 .. X6 are points of the initial mesh as well as the nodal mass Xi in the middle. Nodal mass Xd is the point from the original data set P. (image courtesy of M.-E. Algorri)

Each nodal mass has a dynamic equation of the form:

$$\mu_i \frac{d^2 \boldsymbol{x_i}}{dt^2} + \gamma_i \frac{d \boldsymbol{x_i}}{dt} + \sum_{j=1}^{n} K_{ij}(\boldsymbol{x_i} - \boldsymbol{x_j}) + K_{id}(\boldsymbol{x_i} - \boldsymbol{x_d}) = 0 \qquad (2.7)$$

where $\boldsymbol{x}_i$ is the 3D position of the nodal mass i at time t, μ_i and γ_i are the mass and damping values associated to node i. K_{ij} is the stiffness coefficient of the spring connecting node i to neighboring node j and K_{id} is the stiffness coefficient of the spring connecting node i to its closest point in P $\boldsymbol{x}_d$. The dynamic equations of the nodal massed can be rewritten as:

$$\mu_i \frac{d^2 \boldsymbol{x_i}}{dt^2} + \gamma_i \frac{d \boldsymbol{x_i}}{dt} + (K_d + \sum_{j=1}^{n} K_{ij}) \boldsymbol{x_i} - \sum_{j=1}^{n} K_{ij} \boldsymbol{x_j} = \boldsymbol{x_d} K_{id} \qquad (2.8)$$

and in matrix form:

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{G}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = \boldsymbol{F}_{ext} \qquad (2.9)$$

where $\boldsymbol{M}$ is the mass matrix (diagonal), $\boldsymbol{G}$ is the damping matrix (diagonal), $\boldsymbol{K}$ is the stiffness matrix (contain off-diagonal elements), $\boldsymbol{F}_{ext}$ the external force vector. The adaptive mesh can be seen as as a coupled oscillator with n-degrees of freedom (n = number of the nodal masses in the system). The behavior of this adaptive mesh depends on its dynamic parameters $\boldsymbol{M}$, $\boldsymbol{G}$, $\boldsymbol{K}$. In order to ensure a stable, non-oscillatory behavior of the nodal massed a analysis of the dynamic characteristics is performed.

The solution of the equation (2.9) is the sum of components, a steady-state component and a transient component. The steady-state solution represents the final position of the nodal masses. The form of this solution is found by putting $\frac{d^2 \boldsymbol{x}_i}{dt^2} = \frac{d \boldsymbol{x}_i}{dt} = 0$ in equation (2.8).

$$\boldsymbol{K}\boldsymbol{x} = \boldsymbol{F}_{ext} \qquad (2.10)$$

The transient component of the general solution, is obtained by setting the right-hand side of the equation (2.9) to zero:

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{G}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = 0 \qquad (2.11)$$

By solving both components of the general solution the mesh is warped towards the points. For details see [1]. The process of the reconstruction is illustrated in figure 2.31.

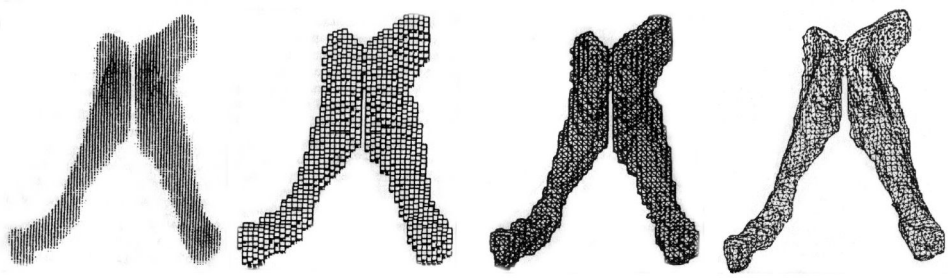


Figure 2.31: An example of surface reconstruction using the approach of M.-E. Algorri and F. Shmitt. In the leftmost figure is the input data set of point. The initial mesh is depicted in the middle figures, where the left one shows the cubes created over the points and the right one contains the smoothed initial mesh that is passed to the warping process. The result of the warping process is depicted on the rightmost figure. (images courtesy of M.-E. Algorri and F. Shmitt)

3 Recent work

The work of Huang, Menq [24] has been used as an initial approach. This algorithm was selected, because it provided linear time complexity, low memory consumption, and because it was a region growing approach. Also this approach can reconstruct surfaces from data sets with slowly changing sampling density and can detect the borders.

In the following text the term *candidate vertex* relates to the term “best point” in the original work.

3.1 Algorithm and data structures

The implemented algorithm slightly modified, and different structures were used, while comparing the implementation with the original paper.

The data structures used in the implementation are as follows, see also figure 3.1:

- The structure of *points* is an array of points, see *P* in the figure 3.1. A uniform subdivision of the point space is used to speedup the point neighborhood lookup, it is not shown in the figure.
- The structure of *triangle mesh* is an array of triangles, where each triangle is formed of three vertices, i.e. indices to the structure of points, see *T* in the figure 3.1. Each triangle also has normal, and the connection between triangles is assured by storing indices to neighbor triangles (over edges). The triangle mesh also has additional data related to vertices, such as vertex normals, because these data are closely related to triangle mesh. The neighbor, normals and additional information are not in the figure.
- The structure of the *border* is currently implemented as a linked list, see *BV* in the figure 3.1. The border is represented by border vertices linked in the order of their connection on the border (*next*, *prev* in the figure) and also linked in the order of their decreasing *candidate values* (*cnext*, *cprev* in the figure). For the term candidate values see below. Each border vertex has information about: estimated normal, neighboring vertices, border triangle it belongs to (*rt* as reference triangle in the figure) and of course candidate vertex that is selected from the neighboring vertices to form new triangle with the edge of the reference triangle (*cv* in the figure) and it's candidate value. There is also some information about the projection of the neighboring points to the tangent plane, in particular for each point in the neighbor an information about its angle in the tangent plane is stored.

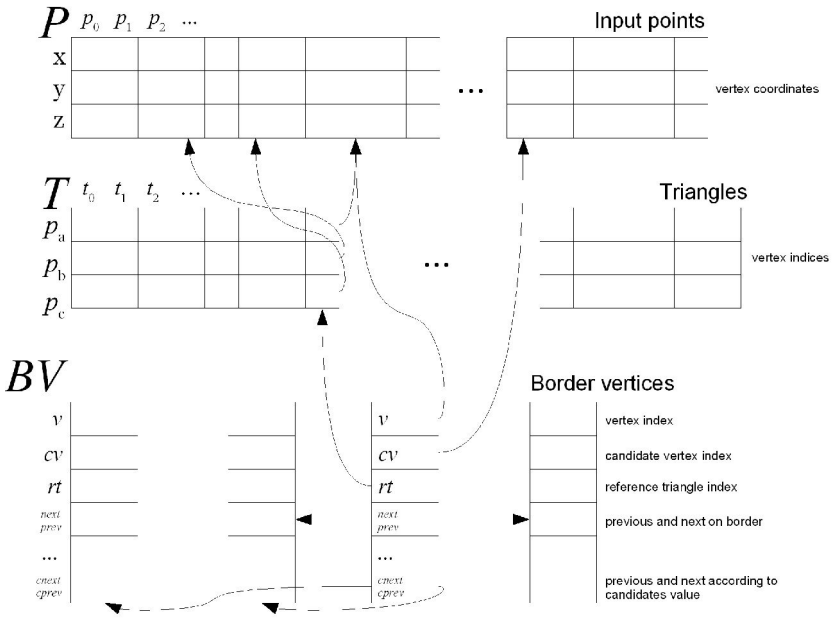


Figure 3.1: The data structures used in the recent implementation of the reconstruction.

The implemented algorithm has three operation instead of four, because linked list has been used to represent the border: *create leaf*, *fill* and *create bridge*. The leaf operation is identical to the vertex joining operation from original algorithm (see figure 3.2a), as well as the fill operation that is identical to the face pasting operation (see figure 3.2b). The operation create bridge handle the two remaining operation: ear attaching and bridge linking (see figure 3.2c). The disadvantage of this change is that the number of separate borders is unknown, however this information is not used throughout the algorithm.

The linked list representing the border is bidirectional, and twice linked: once by the border structure and once by the candidate value. However this structure is not efficient according to speed. While having very large data sets (millions of points) the length of the border (i.e. the complete number of vertices on border) grows incredibly fast and the operations to insert a border vertex or to find a specific border vertex last very long time. This is being solved in the mean time.

The criterion for the selection of the best point is also slightly modified: the point that will form the maximum angle with the edge is selected as the candidate vertex. This

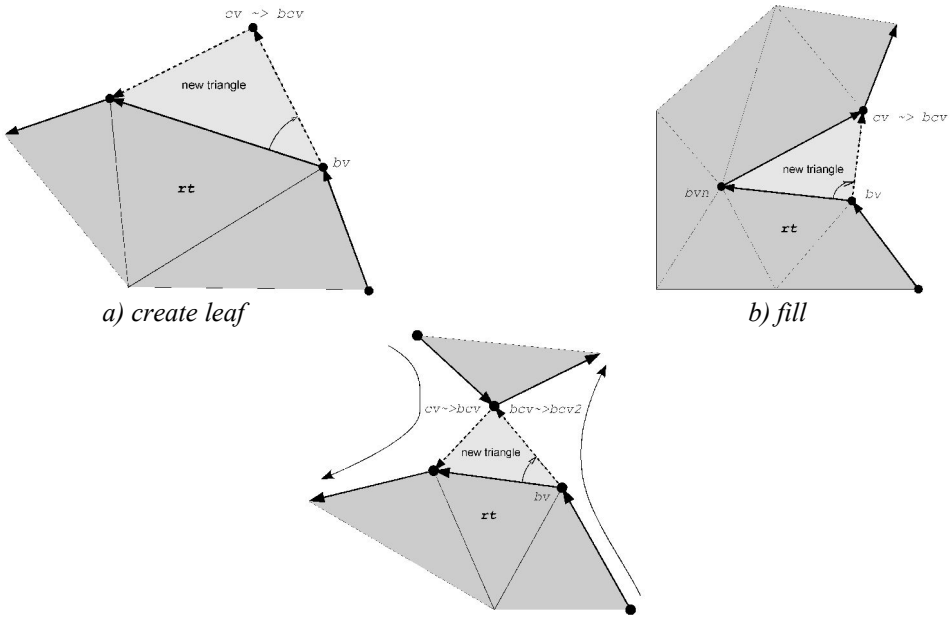


Figure 3.2: Operations used in recent implementation of surface reconstruction. See also Algorithm 2.

maximum angle is related to the candidate value as follows: Let γ be the maximum angle and *cvalue* be the candidate value. Then

$$cvalue = 1 - \cos(\gamma) \tag{3.1}$$

Thus the *cvalue* is from interval (0 ... 2), because the angle γ is from the interval (0 ... π). The bigger the *cvalue* is, the better the candidate is. This criterion has been also used for the altered construction of the initial triangle. The initial triangle is constructed as can be seen in Algorithm 1:

```
1. x1 = get_maximum_point_in_selected_direction(direction);
2. x2 = get_point_nearest_to(x1);
3. x3 = get_point_that_form_biggest_angle_with_edge(x1,x2);
4. create_initial_triangle_from(x1,x2,x3);
5. orient_initial_triangle_to_face(direction);
```

Algorithm 1: Construction of the first triangle/border.

The currently implemented algorithm can be described in short as in the Algorithm 2.

```
1. build_uniform_subdivision_of_the_points();
2. construct_initial_triangle();     /* see above in text */
3. while (exist_border_vertex_with_best_candidate()) {
4.    bv = find_border_vertex_with_best_candidate();
5.    cv = get_best_candidate(bv);
6.    create_new_triangle(bv,cv,bv->next);
7.    switch(which_operation_to_perform(bv,cv)) {
8.    case leaf :
9.       bcv = create_new_border_vertex(cv);
10.      get_neighborhood(bcv);/* use uniform subdivision */
11.      estimate_normal(bcv); /* use neighborhood */
12.      find_best_candidate_for(bcv);
13.      bcv->next = bv->next; bv->next = bcv;
14.      break;
15.   case fill :
16.      bcv = get_border_vertex(cv);
17.      bvn = bv->next;
18.      bv->next = cv;
19.      if(is_removable(bvn))
20.         remove(bvn);
21.      break;
22.   case bridge :
23.      bcv = get_border_vertex(cv);
24.      bcv2 = make_duplicate_border_vertex(bcv);
25.      bcv2->next = bcv->next; bcv->next = bv->next;
26.      bv->next = bcv2;
27.      break;
28.   }
29.   recalculate_candidates(bv,bv->prev,bv->next ... );
30.}
```

Algorithm 2: The recent reconstruction algorithm.

The procedure `recalculate_candidates` on line 29. checks the given border vertex candidate if it's best and correct, otherwise another candidate for the given border vertex is selected. This procedure must be called after the each iteration on the common neighbors of the new border vertex, because the region of possible best candidates formed by the edges was changed.

The function *is_removable* on line 19. checks if the given border vertex (bvn) doesn't have a duplicate created by the third operation (bridge), ie. is not on the border anymore.

Because the border vertices are in queue with the border vertex with the best of the best candidate on the head of this queue, the procedure on the line 3. called `exist_border_vertex_with_best_candidate` simply checks whether there is a border vertex with valid best candidate (i.e. the border vertex is not marked as final (see the original approach)) on the head of the queue.

The function `which_operation_to_perform` on line 7. decides which operation to use: when the vertex is not on the boundary of the mesh use the leaf operation, when the vertex is on the boundary of the mesh and is common neighbor of the current border vertex then use fill operation otherwise use bridge operation.

The procedure `make_duplicate_border_vertex` on line 24. is needed, because each border vertex can have only one area of valid candidates but two borders need to go through this vertex.

The procedure `estimate_normal` on line 11. estimates a normal using principal component analysis and the normal orientation is determined using the normal of the reference triangle *rt*. When the dot product of the estimated normal and the normal of

the triangle is negative, the normal is flipped.

The reconstructed triangular mesh is the first order approximation. Huang & Menq proposed a post-optimization based on curvature that improves the mesh smoothness (according to the estimated curvature). This post-optimization has not yet been implemented.

3.2 Experimental results

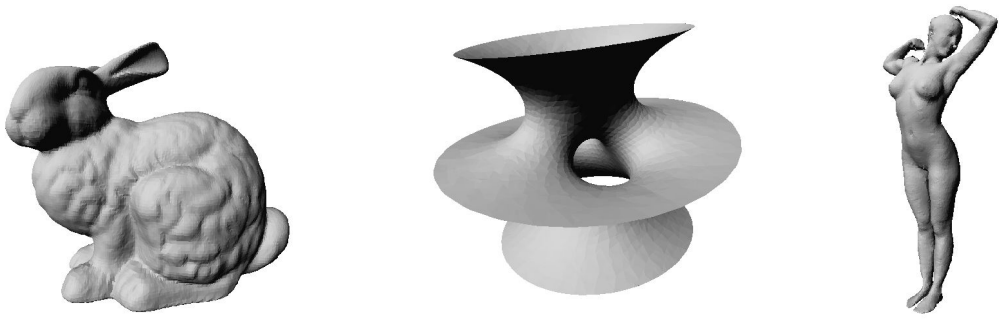
The implemented reconstruction algorithm has been tested on various data, from small artificial data to large real world data. During the testing process, various aspects of the implemented algorithm were observed:

- the influence of the parameter *K* on the resulting mesh and
- time and memory requirements.

Sample of the objects selected for the testing is described in table 1. The data sets are depicted in figure 3.3.

Data set name	***Points (N)***	***Triangles***[4]	***Description***	***see in figure***
`bunny`	35947	69451	Data courtesy of Stanford University	3.3a
`hypersheet`	6752	-	Data courtesy of Hugues Hoppe	3.3b
`woman`	0	-	Data courtesy of Cyberware - cyberware.com	3.3c

Table 1: The data sets used for testing.

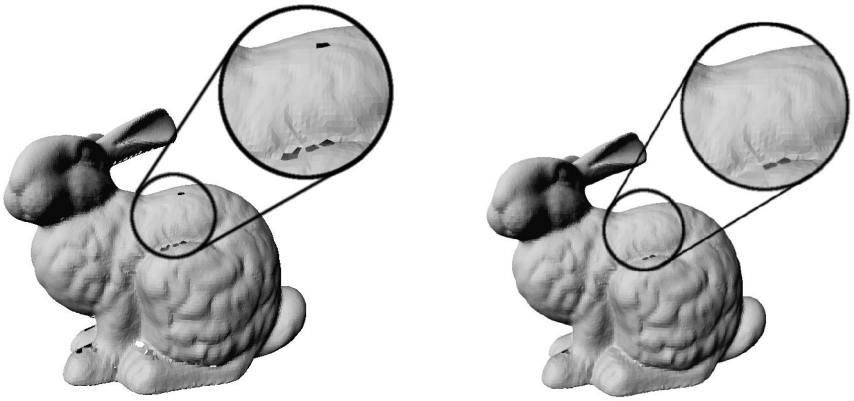


a) `bunny` *b)* `hypersheet` *c)* `woman`

Figure 3.3: Sample of the data sets used for testing.

At first, the influence of the parameter *K* on the reconstructed mesh has been studied. In figure 3.4 is a bunny data set that has been reconstructed by using $K \in \{10, 15$. The holes in the mesh (see the details in circles) indicate under-sampling with respect to the certain value of the parameter *K*. For *K = 20* the bunny data set has been properly reconstructed.

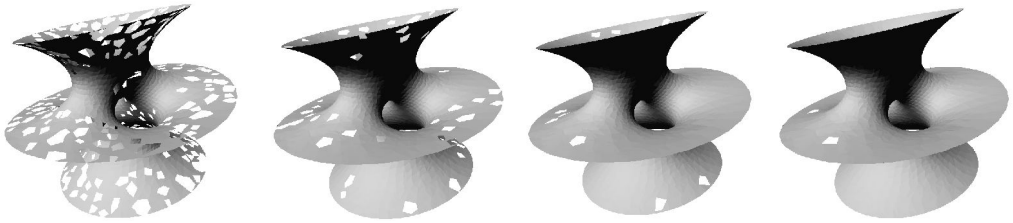
4 if available in the original data set



a) K=10 *b) K=15*

Figure 3.4: The influence of parameter K on the reconstruction – the `bunny` *data set.*

The results of varying *K* for the second data set are depicted on the figure 3.5. In this case *K* has been chosen as $K \in \{10, 15, 20, 25$. Notice that, according to the results in the figures, this second set is sampled worse than the first data set with respect to the used reconstruction algorithm.



a) K=10 *b) K=15* *c) K = 20* *d) K = 25*

Figure 3.5: The influence of the parameter K on the reconstruction - the `hypersheet` *data set.*

It means that the sampling density of the hypersheet data set varies more than the sampling density of the bunny data set and therefore bigger values of *K* must be used to reconstruct the surface properly. The influence of the changes in sampling on the reconstruction process can be reduced by increasing the value of *K*.

The algorithm was also tested on a large data set baba (~ 500 000 points, see in table 1). The results are in figure 3.6.

Also the memory and time efficiency have been observed. The timings of the reconstruction of the used data sets are presented in the table 2.

a) K = 10 *b) K = 15* *c) K = 25*

Figure 3.6: The influence of the parameter K on the reconstruction – the data set `woman`.

Data set	***Approximate timings of the reconstruction (in seconds)***			
	K=10	***K=15***	***K=20***	***K=25***
bunny	27	30	33	36
hypersheet	1,5	1,8	2,2	2,6
woman	4850	5150	5400	5650

Table 2: Time efficiency of the implementation.

There can also be seen the impact of parameter *K* on the duration of the reconstruction process, ie. the algorithm time complexity if O(*N.K*), where N is the number of the triangles reconstructed. However this linear complexity was not achieved because of bad implementation.

In particular, the bad timing results for the data set `woman` are caused by inefficient implementation of the border vertex structure. Currently the algorithm spend most of the time reinserting the border vertex to the structure (procedure `recalculate_candidates`) and while looking for the appropriate border vertex (procedure `get_border_vertex`). This is being solved in the meantime.

The memory used for reconstruction of the data set `woman` (Notice that it has 500 000 points.) didn't overcome the limit of 250MB of memory. This shows that the algorithm is very cheap, in the sense of memory requirements.

The platform that has been used for testing was: PC Intel Pentium III 700Mhz, 1GB of RAM, Microsoft Windows 2000 Professional.

3.3 Conclusion

The implemented algorithm offers many advantages, however there are still some problems. As stated in the opening of this chapter, the advantages of the implemented approach are: linear time complexity (This has not been proved in the implementation, but it is solved in the meantime.), low memory consumption, can reconstruct objects from data sets with slowly changing sampling density (with respect to the parameter K).

However this approach has also disadvantages. The real world data sets contain erroneous samples, that should be detected either by some preprocessing step or by the reconstruction itself. This approach suggests that the input data set is free of erroneous samples. A large value of *K* slows down the reconstruction process. This is a problem related to the sampling. The better the sampling (more uniform) the smaller the *K* can be. However the sampling is changing very much in the real world data set.

The key part of the algorithm is the "best point" selection criterion. The studied criterion has this properties:

- *Part a: the "best point" must lie withing the k-neighborhood of both end points of the edge*. This part causes the edge to be marked as *final* rather than other parts of the criterion. Marking the edge as final leads to the holes forming in the mesh. Small holes are a little problem to be solved, but large holes impose a challenge.
- *Part b: the "best point" must lie withing the area formed by neighboring edges in local tangent plane.* The problematic part is here the evaluation of the local tangent plane, where the principal component analysis is employed to estimate the tangent plane. This may fail in degenerate cases.
- *Part c: the "best point" must form the biggest angle with the edge of all the possible candidates.* This part of criterion doesn't impose any information regarding curvature, normal, ie. the local shape of the mesh.

4 Goals and future work

Each of the reconstruction algorithms has its advantages and disadvantages. Sculpturing methods are very time and memory expensive, but they are the most reliable and robust approaches. They also have provable guarantees when some sampling conditions are met. Volumetric methods provide an approximation of the surface, thus provide smoothing of the surface, but they suffer from the volumetric representation. An uniform volume grid is expensive because the grid size must be large enough to capture every single detail contained in the mesh. The implicit function representation requires some necessary parameters and it is very time and memory consuming. Region growing methods require dense uniform sampling, while they provide linear time complexity and low memory requirements. The advantage and disadvantage of the warping methods is that they require an initial surface that capture topology, and therefore they are a possible post-processing of some approximation techniques.

The implemented method is quite useful for large data sets, however it is pretty slow in meantime and not as robust as expected. Therefore we would like to focus on the following topics in the doctoral thesis:

- derive new criteria where more properties of the local part of the mesh (like curvature) will be observed and thus improve robustness of the method,
- improve the method to be able to process data that are larger than available memory.

5 References

[1] Algorri M.-E., Schmitt F. (1996). Surface Reconstruction from Unstructured 3D Data. *Computer Graphics Forum*, 15(1), pp. 47-60.

[2] Amenta N., Bern M. (1999). Surface Reconstruction by Voronoi Filtering. *Discrete Computational Geometry*, 22(4), pp. 481-504.

[3] Amenta N., Bern M., Kamvysselis M.(1998).A New Voronoi-Based Surface Reconstruction Algorithm,Proceedings of SIGGRAPH 96, Computer Graphics, pp. 415-412.

[4] Amenta N., Choi S., Dey T. K., Leekha N. (2002). A Simple Algorithm for Homeomorphic Surface Reconstruction. *International Journal of Computational Geometry and Applications, 12(1-2), pp. 125-141*.

[5] Amenta N., Choi S., Kolluri R. K. (2000). The Power Crust, Unions of Balls, and the Medial Axis Tranform. *Internation Journal of Computational Geometry andand its Applications(special issue on surface reconstruction)*.

[6] Attali D. (1998). r-Regular Shape Reconstruction from Unorganized Points. *Computational Geometry: Theory and Applications*, 10, pp. 239-247.

[7] Attene M., Spagnuolo M. (2000). Automatic surface reconstruction from point sets in space. *Eurographics*, 19(3).

[8] Bajaj Ch. L., Bernardini F., Chen J., Schikore D. R.(1996).Automatic Reconstruction of 3D CAD Models,Proceedings Theory and Practice of geometric modeling, organized by University of Tubingen, WSI/GRIS.

[9] Bajaj Ch. L., Bernardini F., Xu G. (1997). Reconstructing Surfaces and Functions on Surfaces from Unorganized Three-Dimensional Data. *Algorithmica*, 19, pp. 243-261.

[10] Bernardini F., Bajaj Ch. L., Chen J., Shikore D. R. (1999). Automatic Reconstruction of 3D CAD Models from Digital Scans. *International Journal of Computational Geometry and Applications* 9(4/5). pp. 327-369..

[11] Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G. (1999). The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization andComputer Graphics, 5(4), pp. 349-359*.

[12] Bernardini F., Rushmeier H. (2002). The 3D Model Acquisition Pipeline. *Computer Graphics Forum*, 21(2), pp. 149-172.

[13] Bittar E., Tsingos N., Gascuel M.-P. (1995). Automatic Reconstruction of unstructured 3D Data: Combining a Medial Axis and Implicit Surfaces. *Computer Graphics Forum, 14(3), pp. 457-468*.Proceedings of EUROGRAPHICS '95

[14] Boissonnat J.-D. (1984). Geometric structures for three-dimensional shape reconstruction. *ACM Transaction on Graphics*, 3(4), pp. 266-286.

[15] Boissonnat J.-D., Cazals F. (2000). Smooth Surface Reconstruction via Natual NeighbourInterpolation of Distance Functions. *Proceedings of the 6th European Conferenceon Computer Vision, pp. 588-602.*.

[16] Cignoni, P., Montani, C., Scopigno, R. (1998). DeWall: A fast divide and conquer Delaunay triangulation algorithm. *Computer-Aided Design*, 30, pp. 333-341.

[17] Dey T. K., Giesen J., Hudson J. (2001). Delaunay Based Shape Reconstruction from Large Data. *In proceedings of IEEE Visualization 2001*.

[18] Dey T. K., Leekha N. (1999). Surface Reconstruction Simplified, Manuscript.

[19] Eck M., Hoppe H.(1996).Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type, In Proceedings of SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series. pp. 325-334.

[20] Edelsbrunner H., Mücke E. P. (1994). Three-dimensional Alpha Shapes. *ACM Transactions on Graphics*, 13(1), pp. 43-72.

[21] Franc, M.(2002). *Methods for Polygonal Mesh Simplification*. University of West Bohemia. *State of the Art and Concept of Doctoral Thesis DCSE/TR-2002-01*.

[22] Hoppe H.(1994).Surface Reconstruction from Unorganized Point Clouds,PhD. Thesis. Univeristy of Washington

[23] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W. (1992). Surface Reconstruction from Unorganized Point Clouds. *Computer Graphics*, 26(2), pp. 71-78.Proceeding of SIGGRAPH92.

[24] Huang J., Menq C.-H. (2002). Combinatorial Manifold Mesh Reconstruction and Optimizationfrom Unorganized Points with Arbitrary Topology. *Computer Aided Design*, 34(2), pp. 149-165.

[25] Kobbelt L. P., Bischoff S., Botsch M., Kähler K., Rössl Ch., Schneider R. (2000). Geometric Modeling Based on Polygonal Meshes. *Eurographics Tutorial*.

[26] Krishnamurthy V., Levoy M.(1996).Fitting Smooth Surfaces to Dense Polygon Meshes,Proceedings of SIGGRAPH 96, ACM SIGGRAPH,Computer Graphics Proceedings,Annual Conference Series. pp. 313-324.

[27] Lorensen W. E., Cline H. E. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *SIGGRAPH 87 Conference Proceedings*, 21(4), pp 163-170.

[28] Morse B. S., Yoo T. S., Rheingans P., Chen D. T., Subramanian K. R.,(2001). Interpolating Implicit Surfaces From Scattered Surface Data Using Compactly Supported Radial Basis Functions,Proceedings of International Conference on Shape Modeling and Applications '01,IEEE Computer Society Press . pp. 89-98.

[29] Pulli K., Duchamp T., Hoppe H., McDonald J., Shapiro L., Stuetzle W. (1997). Robust Meshes from Multiple Range Maps. *Proceedings of the International Conferenceon Recent Advances in 3D Digital Imaging and Modeling, Ottawa, Canada, May 1997, pp. 205-211*.

[30] Roth G., Wibowoo E. (1997). An Efficient Volumetric Method for Building Closed TriangularMeshes from 3-D Image and Point Data. *Graphics Interface*, pp. 173-180.

[31] Ruprecht D., Nagel R., Müller H. (1995). Spatial Free Form Deformation With Scattered Data Interpolation Methods. *Computers and Graphics*.19, pp. 63-71.

[32] Szeliski R., Tonnesen D. (1992). Surface Modeling with Oriented Particle Systems. *Computer Graphics*. 26, pp. 185-194.

[33] Taubin, G.(2000).Geometric Signal Processing on Polygonal Meshes

Eurographics State of the Art Report,EUROGRAPHICS' 2000.

6 Publications

[i] Skala V., Kuchař M., Hrádek J. (2001). *Geometry Reconstruction in Rapid Prototyping with Hash Function*. International Conference on Computer Graphics and Imaging CGIM 2001, Honolulu, USA, ISBN 0-88986-303-2, pp.240-245, 2001.

[ii] Hrádek J., Kuchař M., Skala V. (2003). *Hash functions and triangular mesh reconstruction*. Computers & Geosciences. In press.