Interpolation and Approximation of Surfaces from Three–Dimensional Scattered Data Points

Robert Mencl and Heinrich Müller

Informatik VII (Computer Graphics), University of Dortmund, Germany

Abstract

There is a wide range of applications for which surface interpolation or approximation from scattered data points in space is important. Dependent on the field of application and the related properties of the data, many algorithms were developed in the past. This contribution gives a survey of existing algorithms, and identifies basic methods common to independently developed solutions. We distinguish surface construction based on spatial subdivision, distance functions, warping, and incremental surface growing. The systematic analysis of existing approaches leads to several interesting open questions for further research.

1. Introduction

The problem treated in this contribution is

Surfaces from scattered point data.

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

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

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

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

A proper reconstruction of the desired surface in the latter case can only be expected if it is sufficiently sampled. Sufficiency depends on the particular method of surface reconstruction. It might be formulated in form of *sampling theorems* which should give sufficent conditions that can be easily checked. This aspect was neglected in research up to now, and little is known for existing reconstruction algorithms on this aspect. An exception are the works of Attali [2] (which gives a morphology-based sampling theorem at least for the 2D-case) and Bernardini, Bajaj [10].

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

The opposite of insufficient sampling is that the sampling data are unnecessarily dense. This happens in particular if a surface is sampled with uniform density. In that case the sampling density required at fine details of the surface causes too many data points in regions of only minor variation. Several approaches to *data reduction* were proposed in literature [29]. We do not treat this topic here, but only give the hint

that data reduction should consider the power of the reconstruction algorithm expressed in sampling theorems, a fact that also was not explicitly obeyed in the past.

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

The wide range of applications from which the data may emerge implies that the data can have quite different *properties* which may be considered at the solution of the surface interpolation problem. For example, the data may be sampled from surfaces that lie unique over a plane. In that case, a wide range of methods were developed which mainly focus on geometric properties like smoothness of the constructed surface [30].

While the example just mentioned generalizes the problem, reconstruction may become more specific if the surface is captured in multiple samples (multiple view range images) that have to be fused. *Sample fusing* may need data transformation and fitting. We exclude these aspects from further discussion and refer e.g. to [47, 14, 44].

Sample data may contain *additional information on structure*. A typical example are tomographic data. In that case the points on a slice may be already connected by polygonal contour chains. Another example is that normal vectors are available at the data points. These additional informations may give additional hints on the unknown surface which may considered in the construction algorithm. In particular, for interpolation or approximation of contour data, a variety of methods were developed [37]. In the following, no additional structural information is expected.

Finally, the *mathematical and data structural representation* of the derived surface has to be considered. The most common representation is the polygonal or triangular mesh representation. Because the representation by triangular meshes allows to express the topological properties of the surface, and because this is the most difficult sub-problem of surface construction, most known algorithms use this sort of representation. If higher smoothness than just continuity is required, either the parametric or the implicit surface representation may be used. Triangular meshes can be seen as a surface composed by parametrically represented linear surface patches. For surfaces of higher continuity patches of higher order are required. One way to obtain such surfaces is to start from a triangular mesh. For that reason, we have chosen the representation by triangular meshes for this paper, if nothing else is told, and refer to literature for the problem of deriving smooth surfaces, for instance to [17, 24, 20] in which smoothing of surfaces obtained from sample data is particularly emphasized.

The surface construction problem has found considerable interest in the past, and is still an important topic of research. The purpose of this contribution is to find unifying basic methods common to independently developed solutions, coupled with a survey of existing algorithms. The identified basic classes are constructions based on spatial subdivision (section 2), on distance functions (section 3), on warping (section 4), and on incremental surface growing (section 5). In section 6 the aspect is treated that an object represented in a sample data set may consist of several connected components. The systematic analysis of existing approaches leads to several interesting open questions for further research (section 7).

2. Spatial Subdivision

Common to the approaches that can be characterized by "Spatial Subdivision" is that some bounding box of the set P of sampling points is subdivided into disjoint cells. There is a variety of spatial decomposition techniques which were developed for different applications [33]. Typical examples are regular grids, adaptive schemes like octrees, or irregular schemes like tetrahedral meshes. Many of them can also be applied to surface construction.

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

2.1. Surface-Oriented Cell Selection

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

Surface-oriented cell selection:

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

2.1.1. The Approach of Algorri and Schmitt

An example for surface-oriented cell selection is the algorithm of Algorri and Schmitt[1]. For the first step, the rectangular bounding box of the given data set is subdivided by a regular voxel grid. In the second step, the algorithm extracts those voxels which are occupied by at least one point of the sampling set P. In the third step, the the outer quadrilaterals of the selected voxels are taken as a first approximation of the surface. This resembles the cuberille approach of volume visualization [26].

In order to get a more pleasent representation, the surface is transferred into a triangular mesh by diagonally splitting each quadrilateral into two triangles. The cuberille artifacts are smoothed using a depth-pass filter that assigns a new position to each vertex of a triangle. This position is computed as the weighted average of its old position and the position

of its neighbors. The approximation of the resulting surface is improved by warping it towards the data points. For more on that we refer to section 4.2.

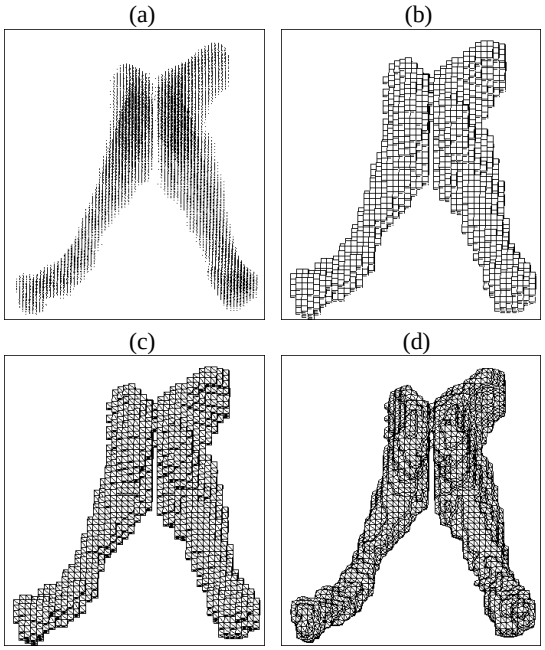

Figure 1: *The steps of the approach of Algorri and Schmitt: the given point set (a), the cuberille face description (b), the triangulated cuberilles (c), and the filtered mesh (d). Picture from Algorri and Schmitt '96* [1]. *© 1996 Eurographics. Reprinted with Permission.*

2.1.2. The Approach of Hoppe et al.

Another possibility of surface-oriented cell selection is based on the distance function approach of Hoppe [28, 29, 27]. The distance function of the surface of a closed object tells for each point in space its minimum signed distance to the surface. Points on the surface of course have distance 0, whereas points outside the surface have positive, and points inside the surface have negative distance. The calculation of the distance function is outlined in section 3.1.1.

The first step of the algorithm again is implemented by a regular voxel grid. The voxel cells selected in the second step are those which have vertices of opposite sign. Evidently, the surface has to traverse these cells. In the third step, the surface is obtained by the marching cubes algorithm of volume visualization [33]. The marching cubes algorithm defines templates of separating surface patches for each possible configuration of the signs of the distance values at the vertices of a voxel cell. The voxels are replaced with these triangulated patches. The resulting triangular mesh separates the positive and negative distance values on the grid.

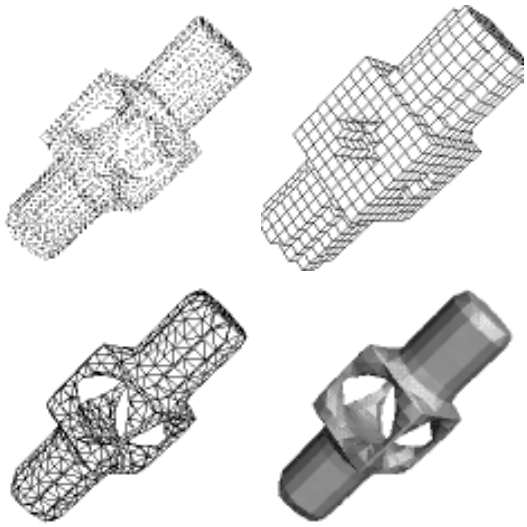

Figure 2: *The approach of Hoppe et al., from left to right: a point set, the nonemtpy voxels of a voxelization of the space, the resulting triangular mesh displayed as wire frame and as shaded surface. Pictures from Hoppe '94* [27]. *Reprinted with Permission.*

A similar algorithm was suggested by Roth and Wibowoo [40]. It differs from the approach of Hoppe et al. in the way the distance function is calculated, cf. section 3.1.2. Furthermore, the special cases of profile lines and multiple view range data are considered besides scattered point data.

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

2.1.3. The Approach of Bajaj, Bernardini et al.

The approach of Bajaj, Bernardini et al. [8] differs from the previous ones in that spatial decomposition is now irregular and adaptive.

The algorithm also requires a signed distance function. For this purpose, a first approximate surface is calculated in a preprocessing phase. The distance to this surface is used as distance function. The approximate surface is calculated using α-solids which will be explained in section 2.2.5.

Having the distance function in hand, the space is incrementally decomposed into tetrahedra starting with an initial tetrahedron surrounding the whole data set. By inspecting the signs of the distance function at the vertices, the tetrahedra traversed by the surface are found out. For each of them, an approximation of the traversing surface is calculated. For this purpose, a Bernstein-Bézier trivariate implicit approximant is used. The approximation error to the given data points is calculated. A bad approximation induces a

further refinement of the tetrahedrization. The refinement is performed by incrementally inserting the centers of tetrahedra with high approximation error into the tetrahedrization. The process is iterated until a sufficient approximation is achieved.

In order to keep the shape of the tetrahedra balanced, an incremental tetrahedrization algorithm is used so that the resulting tetrahedrizations always have the Delaunay property. A tetrahedrization is said to have the *Delaunay property* if none of its vertices lies inside the circumscribed sphere of a tetrahedron [39].

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

In Bernardini et al. [11, 9] an extension and modification of this algorithm is formulated [6, 10]. The algorithm consists of an additional mesh simplification step to reduce the complexity of the mesh represented by the α-solid [7]. The reduced mesh is used in the last step of the algorithm for polynomial-patch data fitting using Bernstein–Bézier patches for each triangle (by interpolating the vertices and normals and by approximating data points in its neighborhood). Additionally, the representation of sharp features can be achieved in the resulting surface.

2.1.4. Edelsbrunner's and Mücke's Alpha-shapes

Edelsbrunner and Mücke [19, 36, 18] also use an irregular spatial decomposition. In contrast to the previous ones, the given sample points are part of the subdivision. The decomposition chosen for that purpose is the Delaunay tetrahedrization of the given set P of sampling points. A tetrahedrization of a set P of spatial points is a decomposition of the convex hull of P into tetrahedra so that all vertices of the tetrahedra are points in P. A tetrahedrization is a *Delaunay tetrahedrization* if none of the points of P lies inside the circumsphere of a tetrahedron. It is well known that each finite point set has a Delaunay tetrahedrization which can be calculated efficiently [39]. This is the first step of the algorithm.

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

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

A problem of this approach is the choice of a suitable α. Since α is a global parameter the user is not swamped with many open parameters, but the drawback is that a variable point density is not possible without loss of detail in the reconstruction.

An example for a reconstruction of a body is shown in Figure 3. If α is too small, gaps in the surface can occur, or the surface may become fragmented.

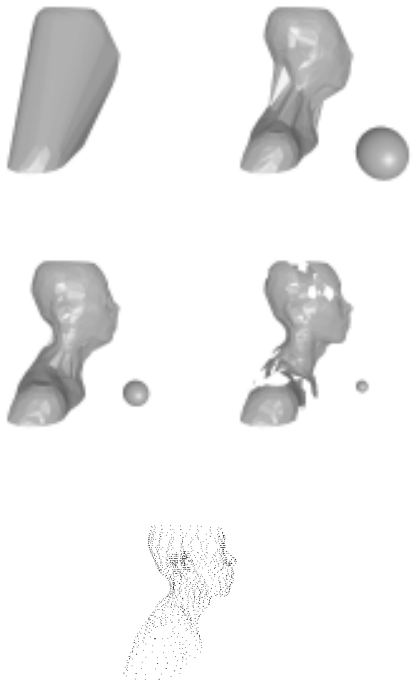

Figure 3: *Edelsbrunner's and Mücke's α-shapes applied to a sampled bust. Different α, visualized by the corresponding α-balls, were used. The bottom figure shows the given data set, the upper left one its convex hull. The pictures were generated with original α-shape software* [19, 36, 18].

Guo et al. [25] also make use of α–shapes for surface reconstruction but they propose a so–called *visibility* algorithm for extracting those triangles out of the α–shape which represent the simplicial surface.

2.1.5. Attali's Normalized Meshes

In the approach of Attali [2], the Delaunay tetrahedrization is also used as a basic spatial decomposition. Attali introduces so-called normalized meshes which are contained in the Delaunay graph. It is formed by the edges, faces and tetrahedra whose dual Voronoi element intersects the surface of the object.

In two dimensions, the normalized mesh of a curve c consists of all edges between pairs of points of the given set P of

sampling points on c which induce an edge of the Voronoi diagram of P that intersects c. The nice property of normalized meshes is that for a wide class of curves of bounded curvature, the so-called r-regular shapes, a bound on the sampling density can be given within which the normalized mesh retains all the topological properties of the original curve.

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

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

To overcome this problem, two different heuristics for filling the gaps in the surface structure were introduced.

The first heuristic is to triangulate the border of a gap in the triangular mesh by considering only triangles contained in the Delaunay tetrahedrization.

The second heuristic is volume-based. It merges Delaunay tetrahedra to build up the possibly different solids represented in the point set. The set of mergeable solids is initialized with the Delaunay tetrahedra and the complement of the convex hull. The merging step is performed by processing the Delaunay triangles according to decreasing diameters. If the current triangle separates two different solids in the set of mergable solids, they are merged if the following holds:

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

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

2.1.6. Weller's approach of Stable Voronoi Edges

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

It turns out that for intuitively reasonably sampled curves in the plane, the stable edges usually are the edges connecting two consecutive sampling points on the curve, whereas the edges connecting non-neighboring sampling points are instable. The stability of an edge can be checked in time $O(\#\text{Voronoi neighbors})$ [50].

The extension of this approach to 3D-surfaces shows that large areas of a surface can usually be reconstructed correctly, but still not sufficiently approximated regions do exist. This resembles the experience reported by Attali [2], cf. section 2.1.5. Further research is necessary in order to make stability useful for surface construction.

2.2. Volume-Oriented Cell Selection

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

Volume-oriented cell selection:

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

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

Most implementations of volume-oriented cell selection are based on the Delaunay tetrahedrization of the given set P of sampling points. The algorithms presented in the following differ in how volume-based selection is performed. Some algorithms eliminate tetrahedrons expected outside the desired solid, until a description of the solid is achieved [13, 31, 48]. Another methodology is the use of the Voronoi diagram to describe the constructed solid by a "skeleton" [43, 2].

2.2.1. Boissonnat's Volume-Oriented Approach

Boissonnat's volume-oriented approach starts with the Delaunay triangulation of the given set P of sampling points. From this triangulation of the convex hull, tetrahedra having particular properties are successively removed. First of all, only tetrahedra with *two faces, five edges and four points* or *one face, three edges and three points* on the boundary of the current polyhedron are eliminated. Because of this elimination rule only objects without holes can be reconstructed, cf. Figure 4. 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. This decision value is useful because flat tetrahedra of the Delaunay tetrahedrization usually tend to be outside the object and cover areas of higher detail. The algorithm stops if all points lie on the surface, or if the deletion of the tetrahedron with highest decision value does not improve the sum taken over the decision values of all tetrahedra incident to the boundary of the polyhedron.

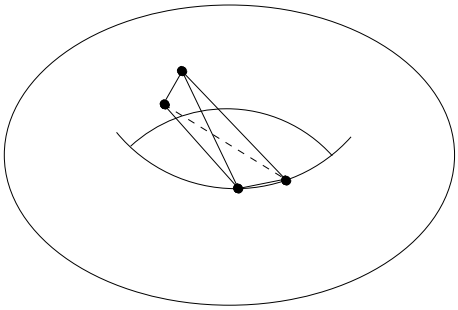

Figure 4: *Boissonnat's volume-oriented approach. An example for a tetrahedron which cannot be removed by the elimination rule of Boissonnat. The tetrahedron in the hole of the torus has four faces on the boundary.*

2.2.2. The Approach of Isselhard, Brunnett, and Schreiber

The approach of [31] is an improvement of the volume-oriented algorithm of Boissonnat [13]. While Boissonnat cannot handle objects with holes, the deletion procedure of this approach is modified so that construction of holes becomes possible.

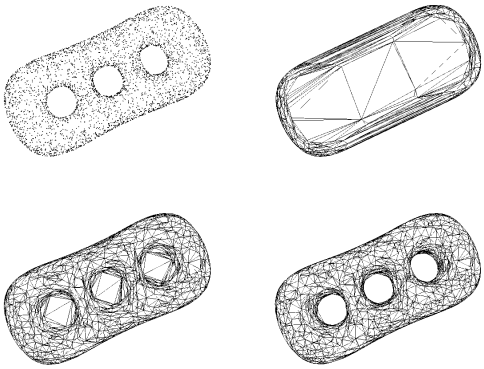

Figure 5: *The approach of Isselhard, Brunnett, and Schreiber. From left to right: The point set, the convex hull, an intermediate step during the process of tetrahedra elimination, and the reconstruction result* [31]. *Courtesy of Arbeitsgruppe CAD und Algorithmische Geometrie, Fachbereich Informatik, University of Kaiserslautern, Germany. Reprinted with Permission.*

As before, the algorithm starts with the Delaunay triangulation of the point set. An incremental tetrahedron deletion procedure is then performed on tetrahedra at the boundary of the polyhedron, as in Boissonnat's algorithm. The difference is, that more types of tetrahedron can be removed in order to being able to reconstruct even object with holes, remember the configuration depicted in Figure 4. The additionally allowed types of tetrahedra are those with *one face and four vertices* or *three faces* or *all four faces* or on the current surface provided that no point would become isolated through their elimination.

The elimination process is controlled by observing an *elimination function*. The elimination function is defined as the maximum decision value (in the sense of Boissonnat) of the remaining removable tetrahedra. In this function, several significant jumps can be noticed. One of these jumps is expected to indicate that the desired shape is reached. In practice, the jump before the stabilization of the function on a higher level is the one which is taken. This stopping point helps handling different point densities in the point set which would lead to undesired holes through the extended type set of removable tetrahedra in comparison to Boissonnat's algorithm [13].

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

An example point set and the deletion process is depicted in Figure 5.

2.2.3. The γ–indicator Approach of Veltkamp

To describe the method of Veltkamp [48, 49] some terminology is required. A γ*-indicator* is a value associated to a sphere through three boundary points of a polyhedron which is positive or negative, cf. Figure 6 for an illustration of the 2D-case. Its absolute value is computed as $1 - \frac{r}{R}$, where r is the circle for the boundary triangle and R the radius of the boundary tetrahedron. It is taken to be negative if the center of the sphere is on the inner side and positive if the center is on the outer side of the polyhedron. Note, that the γ-indicator is independent of the size of the boundary triangle (tetrahedron, respectively). Therefore, it adapts to areas of changing point density. A removable face is a face with positive γ–indicator value.

The first step of the algorithm is to calculate the Delaunay tetrahedrization.

In the second step, a heap is filled with removable tetrahedra which are sorted according to their γ–indicator value. The removable tetrahedra are of the same boundary types as in Boissonnat's volume-oriented approach [13]. The tetrahedron with the largest γ-indicator value is removed and the

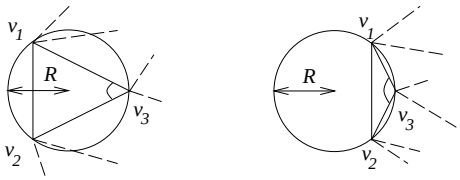

Figure 6: *Two cases for the γ-indicator value, in the 2D-case. Picture from Veltkamp '94* [48]. *© 1994 Springer Verlag. Reprinted with Permission.*

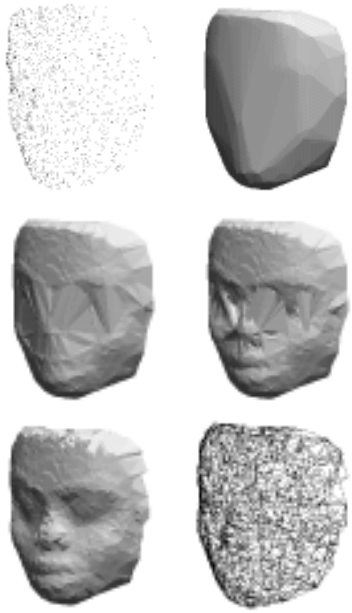

Figure 7: *The approach of Veltkamp. From left to right: The given point set, its convex hull, two intermediate stages of construction, the final result, and the resulting mesh rendered as wire frame. Pictures from Veltkamp '94* [48]. *© 1994 Springer Verlag. Reprinted with Permission.*

boundary is updated. This process continues until all points lie on the boundary, or there are no further removable tetrahedra.

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

Some intermediate stages during the construction of a surface are displayed in Figure 7.

2.2.4. The Approach of Schreiber and Brunnett

The approach of Schreiber and Brunnett [42, 43] uses properties of the Voronoi diagram of the given point set for tetrahedra removal. The *Voronoi diagram* of a point set P is a

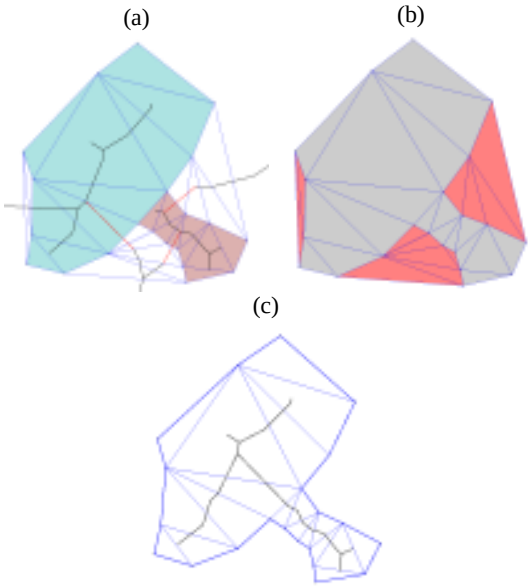

Figure 8: *The approach of Schreiber and Brunnett. From left to right: (a) The desired polygon (filled) and Delaunay triangulation of the given points with an inner MST (drawn fat), (b) the classification into inner and outer regions, (c) and the reconstructed polygon. The pictures have been generated for this paper by the Arbeitsgruppe CAD und Algorithmische Geometrie, Fachbereich Informatik, University of Kaiserslautern, Germany* [42, 43].

partition of the space in regions of nearest neighborhood. For each point $\mathbf{p}$ in P, it contains the region of all points in space that are closer to $\mathbf{p}$ than to any other point of P. It is interesting to note that the Voronoi diagram is dual to the Delaunay tetrahedrization of P. For example, each vertex of the Voronoi diagram corresponds to the center of a tetrahedron of the tetrahedrization. Edges of the Voronoi diagram correspond to neighboring faces of the tetrahedra dual to its vertices. The same observation holds for Voronoi diagrams in the plane that are used in the following for the explanation of the 2D-version of the algorithm.

In the first step, the Delaunay triangulation and the dual Voronoi diagram of P is determined. The second step, the selection of tetrahedra, uses a minimum spanning tree of the Voronoi graph, cf. Figure 8 (a). The *Voronoi graph* is the graph induced by the vertices and edges of the Voronoi diagram. A *minimum spanning tree* (MST) of a graph is a subtree of the graph which connects all vertices and has minimum summed edge length. Edge length in our case is the Euclidean distance of the two endpoints of the edge.

In the second step, a pruning strategy is applied to it which possibly decomposes it into several disjoint subtrees, cf. Figure 8 (a). Each subtree represents a region defined by the union of the triangles dual to its vertices.

Two pruning rules have been developed for that purpose:

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

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

The resulting regions are then distinguished into inside and outside, cf. Figure 8 (b). In order to find the inside regions, we add the complement of the convex hull as further region to the set of subtree regions. The algorithm starts with a point on the convex hull which is incident to exactly two regions. The region different from the complement of the convex hull is classified "inside". Then the label "inside" is propagated to neighboring regions by again considering points that are incident to exactly two regions.

After all regions have been classified correctly, the boundary of the constructed shape is obtained as the boundary of the union of the region labeled "inside", cf. Figure 8 (c).

An adaption of this method to three dimensions is possible.

2.2.5. The α-solids of Bajaj, Bernardini et al.

Bajaj, Bernardini et al. [8, 6, 10, 11] calculate so-called α-*solids*. While α-shapes are computed by using eraser spheres at every point in space, the eraser spheres are now applied from outside the convex hull, like in Boissonnat's approach [13]. To overcome the approximation problems inherent to α-shapes a re-sculpturing scheme has been developed. Re-sculpturing roughly follows the volumetric approach of Boissonnat in that further tetrahedra are removed. This goal is to generate finer structures of the object provided the α-shape approach has correctly recognized the larger structures of the object.

3. Surface Construction with Distance Functions

The distance function of a surface gives the shortest distance of any point in space to the surface. For closed surface the distances can be negative or positive, dependent on whether a point lies inside or outside of the volume bounded by the surface. In sections 2.1.2 and 2.2.5, we have already described an algorithm which uses the distance function for the purpose of surface construction. There the question remained open how a distance function can be calculated from the given set P of sample points. Solutions are presented in the next subsection.

Another possibility of calculating a distance function is to construct a surface to the given set P of data points and take the distance to this surface. The idea behind that is that this distance function may be used to get a better surface, for instance a smooth surface as in [8].

Besides marching cubes construction of surfaces as explained in section 2.1.2, distance plays a major role in construction of surfaces using the medial axis of a volume. The medial axis consists of all points inside the volume for which the maximal sphere inside the volume and centered at this point does not contain the maximal sphere of any other point. Having the medial axis and the radius of the maximum sphere at each of its points, the given object can be represented by the union taken over all spheres centered at the skeleton points with the respective radius. An algorithm for surface construction based on medial axes is described in section 3.2.

3.1. Calculation of Distance Functions

3.1.1. The Approach of Hoppe et al.

Hoppe et al. [28, 27] suggest the following approach. At the beginning, for each point $\mathbf{p}_i$ an estimated tangent plane is computed. The tangent plane is obtained by fitting the best approximating plane in the least square sense [16] into a certain number k of points in the neighborhood of $\mathbf{p}_i$. In order to get the sign of the distance in the case of close surfaces, a consistent orientation of neighboring tangent planes is determined by computing the *Riemannian graph*. The vertices of the Riemannian graph are the centers of the tangent planes which are defined as the centroids of the k points used to calculate the tangent plane. Two tangent plane centers $\mathbf{o}_i, \mathbf{o}_j$ are connected with an edge (i, j) if one center is in the k-neighborhood of the other center. By this construction, the edges of the Riemannian graph can be expected to lie close to the sampled surface. Each edge is weighted by 1 minus the absolute value of the scalar product between normals of the two tangent plane centers defining the edge. The orientation of the tangent planes is determined by propagating the orientation at a starting point, by traversing the minimum spanning tree of the resulting weighted Riemannian graph.

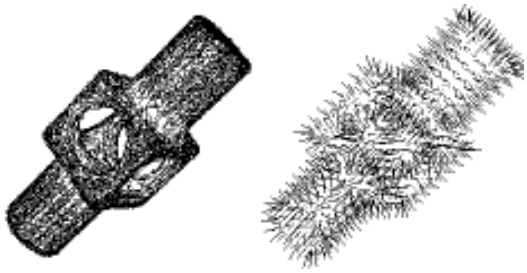

Figure 9: *Left: The Riemannian Graph. Right: The distance vectors from the vertices of the surrounding voxel grid. Pictures from Hoppe '94* [27]*. Reprinted with Permission.*

Using the tangent plane description of the surface and their correct orientations, the signed distance is computed by first determining the tangent plane center nearest to the query point. The distance between the query point and its

projection on the nearest tangent plane. The sign is obtained form the orientation of the tangent plane.

3.1.2. The Approach of Roth and Wibowoo to Distance Functions

The goal of the algorithm of Roth and Wibowoo [40] is to calculate distance values at the vertices of a given voxel grid surrounding the data points. The data points are assigned to the voxel cells into whcih they fall. An "outer" normal vector is calculated for each data point by finding the closest two neighboring points in the voxel grid, and then using these points along with the original point to compute the normal.

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

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

3.2. Bittar's et al. Surface Construction by Medial Axes

The approach of Bittar et al. [12] consists of two steps, the calculation of the medial axis and the calculation of an implicit surface from the medial axis.

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

The desired surface is calculated by distributing centers of spheres on the medial, cf. Figure 10 (a). The radius of a sphere is equal to the distance assigned to its center on the medial axis. For each sphere, a field function is defined which allows to calculate a scalar field value for arbitrary point in space. A field function of the whole set of spheres is obtained as sum of the field functions of all spheres. The

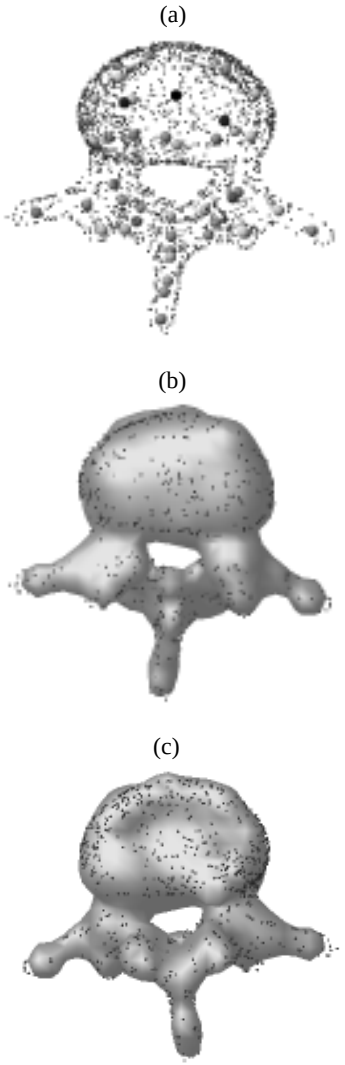

Figure 10: *Surface Construction by Medial Axes. (a) The point set with medial axis points selected according to the search strategy. (b) The constructed implicit surface. (c) The object from behind. Picture: Courtesy of Bittar et al.* [12]. *© 1995 Eurographics. Reprinted with Permission.*

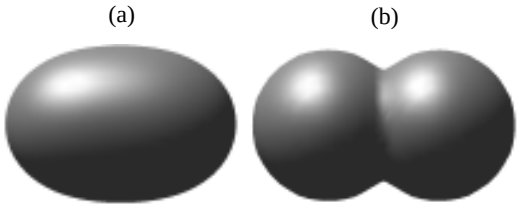

Figure 11: *A soft field function melts two spheres together (a) while a sharp field function preserves detail (b). Picture from Bittar et al. 95* [12]. *© 1995 Eurographics. Reprinted with Permission.*

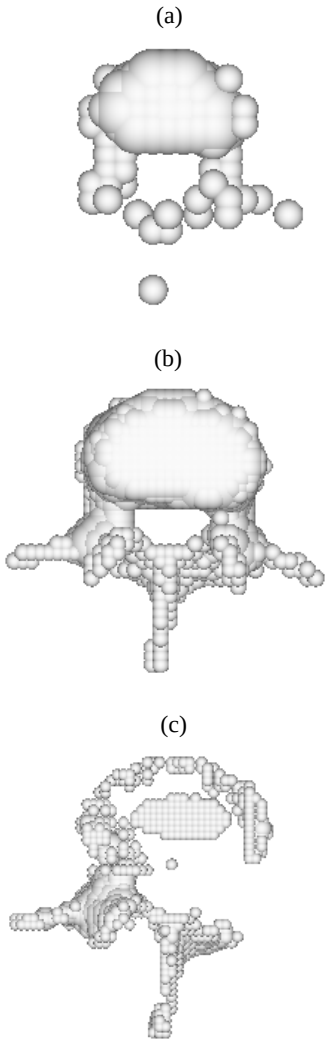

Figure 12: *Surface Construction by Medial Axes. (a) Low precision. (b) Mid precision. (c) The precision is too high. Picture from Bittar et al. 95* [12]. *© 1995 Eurographics. Reprinted with Permission.*

implicit surface is defined as an iso-surface of the field function, that is, it consists off all points in space for which the field function has a given constant value, cf. Figure 10 (b),(c).

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

The shape of the resulting surface is strongly influenced by the type of field function. For example, a *sharp* field function preserves details while a *soft* function smoothes out the details, cf. Figure 11. Also the connectness of the resulting solid can be influenced by the shape function cf. Figure 12.

Because of the voxelization, a crucial point is tuning the resolution of the medial axis. If the resolution of the axis is low, finer details are not represented very accurately. The display of the surface detail is improved if the resolution is increased but can also tend to disconnect parts of the surface if the resolution is higher than the sample density at certain regions, cf. Figure 12.

A result of this algorithm is shown in Figure 10.

4. Surface Construction by Warping

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

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

Several methods of describing deformable surfaces were developed in the past. Muraki suggested a "*blobby model*" to approximate 2.5 D range images [38]. Terzopoulos, Witkin and Kass [46, 15] made use of *deformable superquadrics* which have to fit the input data points.

Miller et al. [32] extract a topologically closed geometric model from a volume data set. The algorithm starts with a simple model that is already topologically closed and deforms the model on a set of constraints, so that the model grows or shrinks to fit the object within the volume while maintaining it closed and a locally simple non-self-intersecting polyhedron that is either embedded in the object or surrounds the object in the volume data representation. A function is associated with every vertex of the polyhedron that associates costs with local deformation adherent to properties of simple polyhedra, and the relationship between noise and feature. By minimizing these constraints, one achieves an effect similiar to inflating a balloon within a container or collapsing a piece of shrink wrap around the object.

A completely different approach to warping is modeling with *oriented particles* suggested by Szeliski and Tonnesen [45]. Each particle owns several parameters which are updated during the modeling simulation. By modeling the interaction between the particles themselves the surface is being modeled using forces and repulsion. As an extension Szeliski and Tonnesen describe how their algorithm can be extended for automatic 3D reconstruction. At each sample location one particle with appropriate parameters is generated. The gaps between the sample points (particles, respectively) are filled by growing particles away from isolated

points and edges. After having a rough approximation of the current surface the other particles are rejusted to smooth the surface.

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

4.1. Spatial Free Form Warping

The idea of spatial free-form warping is to deform the whole space in which an object to be warped is embedded in, with the effect that the object is warped at the same time. Space deformation is defined by a finite set of displacement vectors consisting of pairs of initial and target point, from which a spatial displacement vector field is interpolated using a scattered data interpolation method. There is a huge number of scattered data interpolation methods known in literature, cf. e.g. [30]. Among them that one can be chosen that yields the most reasonable shape for the particular field of application.

The resulting displacement vector field tells for each point in space its target point. In particular, if the displacement vector field is applied to all vertices of the initial mesh, or of a possibly refined one, the mesh is warped towards the given data points [41].

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

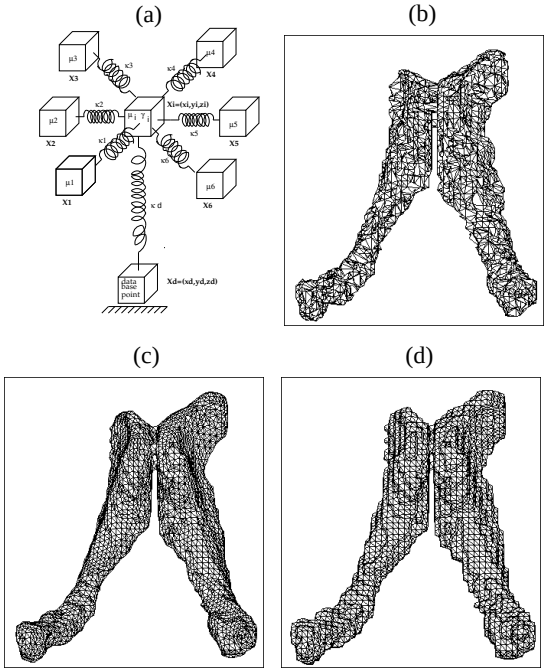

Figure 13: *(a) Illustration of the spring model used in the dynamic model warping approach of Algorri and Schmitt. (b)-(d) show the resulting meshes for three different parameter values. Picture from Algorri and Schmitt '96* [1]. © *1995 Eurographics. Reprinted with Permission.*

4.2. The Approach of Algorri and Schmitt

The idea of Algorri and Schmitt [1] is to translate given approximate triangular mesh into a physical model, cf. Figure 13. The vertices of the mesh are interpreted as mass points. The edges are replaced with springs. Each nodal mass of the resulting mesh of springs is attached to its closest point in given set P of sampling points by a further spring. The masses and springs are chosen so that the triangular mesh is deformed towards the data points. Figure 13(b)–(d) shows the resulting meshes for three different parameter values on the data of Figure 1.

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

4.3. Kohonen Feature Map Approach of Baader and Hirzinger

The Kohonen feature map approach of Baader and Hirzinger [4, 5, 3] can be seen as another implementation of the idea of

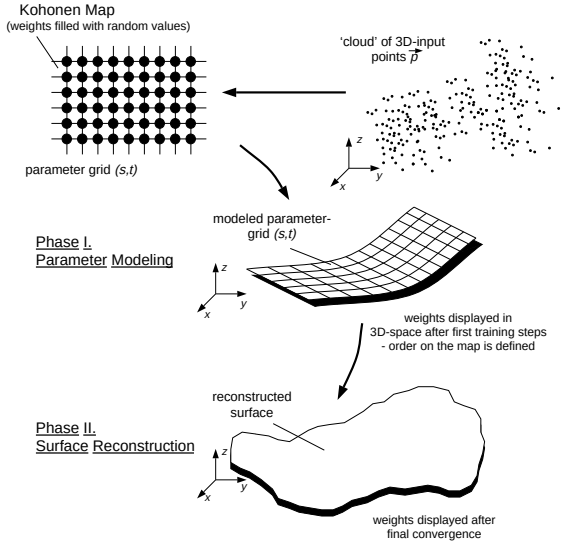

Figure 14: *Arranging the Kohonen feature map to a point set in three dimensions. Picture: Courtesy of Baader and Hirzinger* [5]. © *1993 IEEE. Reprinted with Permission.*

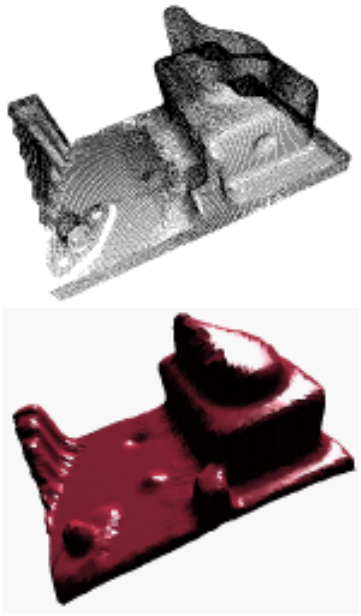

Figure 15: *Top: The point set. Bottom: The result of the Kohonen feature map approach. Pictures from Baader '95* [3]. *© 1995 VDI Verlag. Reprinted with Permission.*

surface construction by warping. Kohonen's feature map is a two-dimensional array of units (neurons), cf. Figure 14. Each unit u_j has a corresponding weight vector $\vec{w}_j$. In the beginning these vectors are set to normalized random values (of length equal to 1). During the reconstruction or training process the neurons are fed with the input data which affects their weight vectors (which resemble their position in three-space). Each input vector $\vec{i}$ is presented to the units j which produce output o_j of the form

$$o_j = \vec{w}_j \cdot \vec{i},$$

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

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \varepsilon_j \cdot (\vec{i} - \vec{w}_j(t))$$

Note that after this update the weight vectors have to be normalized again. The value $\varepsilon_j = \eta \cdot h_j$ contains two values, the learning rate η and the neighborhood relationship h_j. Units far away from the center of excitation are only slightly changed.

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

The combination of this normal and reverse training completes the algorithm of Baader and Hirzinger and has to be used in the training of the network.

A result is depicted in Figure 15.

5. Incremental Surface-Oriented Construction

The idea of incremental surface-oriented construction is to build-up the interpolating or approximating surface directly on surface-oriented properties of the given data points. This can be done in quite different manner.

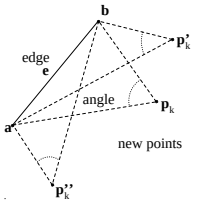

Figure 16: *Point* $\mathbf{p}_k$ *sees edge* **e** *under the largest angle.*

For example, surface construction may start with an initial surface edge at some location of the given point set P, connecting two of its points which are expected neighboring on the surface. The edge is successively extended to a larger surface by iteratively attaching further triangles at boundary edges of the emerging surface. The surface-oriented algorithm of Boissonnat explained in the first subsection may be assigned to this category.

Another possibility is to start with a global wire frame of the surface, in order to fill it iteratively to a complete surface. This is the idea of the approach of Mencl and Müller described in section 5.2.

5.1. Boissonat's Surface-Oriented Approach

Boissonnat's surface oriented contouring algorithm [13] usually starts at the shortest connection between two points of the given point set P. In order to attach a new triangle at this edge, and later on to other edges on the boundary, a locally estimated tangent plane is computed based on the points in the neighborhood of the boundary edge. The points in the neighbourhood of the boundary edge are then projected onto the tangent plane. The new triangle is obtained by connecting one of these points to the boundary edge. That point is taken which maximizes the angle between at its edges in the

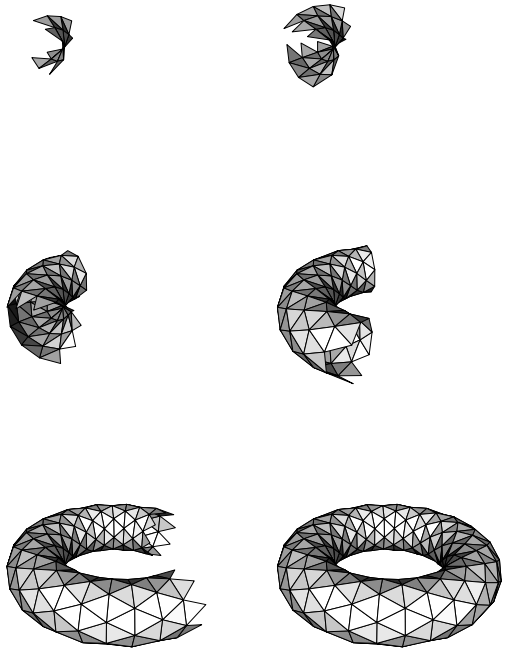

Figure 17: *This figure shows the behavior of a contouring algorithm like Boissonnat's [13] during the reconstruction of a torus. The picture sequence was not reconstructed by the original software (which was not available).*

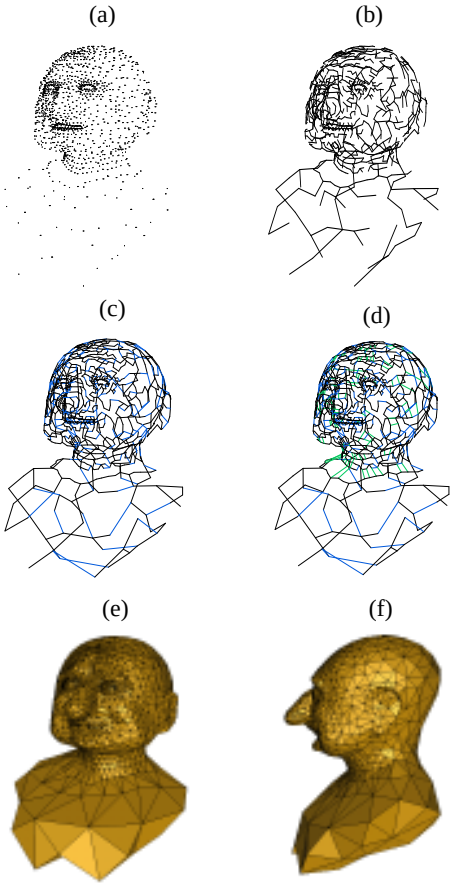

Figure 18: *The approach of Mencl and Müller. From left to right: (a) the initial point set, (b)-(d) three different steps during the graph extension, and the final results (e),(f) seen from two different view points. Note the high changes of point density in the point set. Point set: Courtesy of Academy of Media Arts in Cologne, Germany.*

new triangle, that is, the point sees edge boundary edge under the maximum angle, cf. Figure 16. The algorithm terminates if there is no free edge available any more. The behavior of this algorithm can be seen in Figure 17.

5.2. The Approach of Mencl and Müller

The solution of Mencl and Müller consists of seven main steps [34, 35]:

1. The computation of the EMST (*Euclidean minimum spanning tree*) of the point set.
2. Extension of the graph at leaf points of the EMST.
3. Recognition of features.
4. Extraction of different objects out of the graph.
5. Connection of features of the same kind.
6. Connection of associated edges in the graph.
7. Filling the wire frame with triangles.

The first two steps are designed to build up an initial *surface description graph* (SDG). This is performed by computing the EMST (*Euclidean minimum spanning tree*) and an graph extension step afterwards, cf. Figure 18 (b),(c). Next, a feature recognition is performed to gain necessary information considering the possible structure of the surface in the third step. As in object recognition of raster images Mencl and Müller consider features to be regions with special information about the objects structure like paths, edges, point rings and so on. After that, these feature areas are disconnected and/or connected according to certain rules to have a proper description of the objects in the point set (step 4 and 5). In the last step before the triangle filling procedure, the so far computed graph is extended more by connecting associated edges in the graph under consideration of certain constraints, cf. Figure 18 (d). Finally, the triangles are filled into this surface description graph by using a rule system to assure a resulting surface with high accuracy (Figure 18 (e),(f)).

As a main concept, Mencl and Müller introduce the concept of feature recognition and clustering to improve the accuracy of the surface description graph according to the as-

sumed surface of the object [35]. The idea is the possibility to integrate different kind of recognition algorithms in the main algorithm while maintaining the structural consistency of the SDG.

In contrast to many other methods this approach returns a piecewise linear surface which interpolates exactly the input point set. The algorithm can handle point sets with high changes in point density. This makes it possible to describe objects with only the least necessary amount of points since it is not necessary to oversample areas with low local curvature. The reconstruction of sharp edges in artificial or synthetic objects can be done properly as well as the reconstruction of non-orientable surfaces like Möbius strips, for example.

6. Clustering

It may happen that more than one connected shape is represented in a sample data set. Figure 19 (a) shows an example. In that case, most of the methods discribed up to now may have troubles. The difficulty can be overcome by *segmenting* or *clustering* the sample point set P into subsets of points which are likely to belong to the same component.

6.1. The Approach of Fua and Sander

The following approach of Fua and Sander [21, 22, 23] is an example of how clustering can be performed. It consists of three steps.

In the first step, a quadric surface patch is iteratively fitted around every data point, and the data point is moved onto the surface patch. One additionally effect of this step besides yielding a set of local surfaces is that a smoothing of the given sample data is performed. Figures 19 (a),(b) show a set of sample points, and Figure 19 (c) the result of smoothing.

When smoothing is done, the data points still form an irregular sampling of the underlying surface. In the second step, the sample points together with their local surface patches are moved on positions on a regular grid. Figure 19 (d) depicts the result.

In the third step, a surface-oriented clustering is performed. A graph is calculated whose vertices are the corrected sample points of the previous step. An edge is introduced between two vertices if the quadrics assigned with them are similiar. A measure of similarity and a threshold are defined for that purpose. The connected components of the graph define the clusters of the surface in the data set, cf. Figure 19 (e),(f).

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

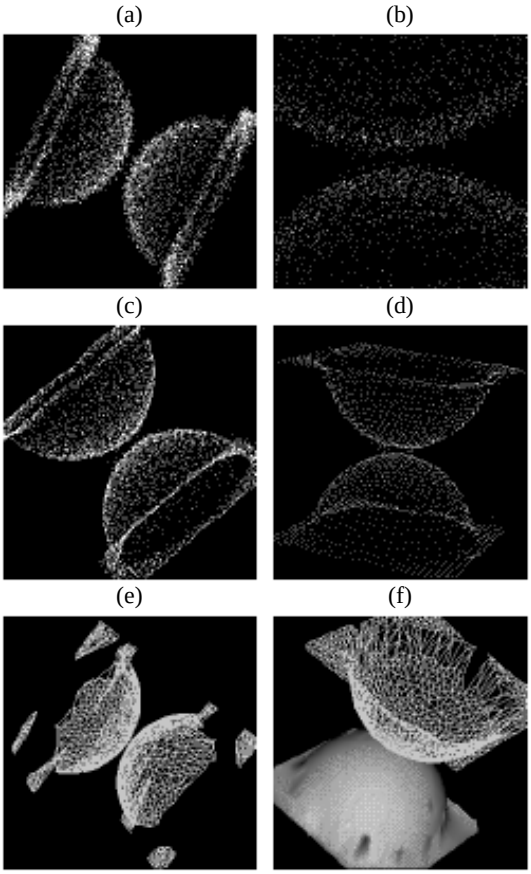

Figure 19: *The approach of Fua and Sander. From left to right and top to down: (a) Two noisy hemispheres. (b) A zoom into the scene. (c) Smoothened data of the spheres after several iterations. (d) Resampled points on the estimated surface. (e),(f) Two possible segmentations of the data points according to the segmentation parameter. The 2D Delaunay triangulation in the tangent plane have been used in both cases to generate the triangular mesh. In (f), one the of the spheres is shaded. Pictures from Fua and Sander [22].*

6.2. The Approach of Mencl and Müller according to Clustering

Clustering in the approach of Mencl and Müller [34, 35] (cf. section 5.2) is performed in three consecutive stages.

First, all edges of the surface description graph are investigated according to their length. If the length of an edge exceeds a certain local density value (determined by the local point density at each of its end points) the edge will be deleted provided that it does not connect any feature regions. If a single edge connects certain feature regions, the edge will remain in the graph. Features often imply a certain struc-

ture of a surface and a precise consideration in the next stage is necessary.

Second, all edges connecting feature regions are considered in more detail. Depending on the structural information contained in these features it is determined whether the connecting edges contradict to the assumed surface structure and are deleted if necessary.

Third, the neighborhood of all features is examined. The best connection (a new edge) between neighboring features is computed and added to the surface description graph if it is appropriate according to the feature information.

7. Further Developments

The difficulty with the surface construction problem is that a unique or the desired solution cannot be obtained from the given set of sampling points. In the case of data analysis, finding possible and reasonable solutions just is the goal of surface construction. In the case of reconstruction of a known surface, the data may be not sufficiently dense so that a good approximating surface can be constructed. The intention of the various methods presented here is to find solutions that cover as many critical cases as possible, usually by clever ad-hoc heuristics. A more general framework, possibly including interactivity seems necessary in order to get a powerful system of surface construction. In the following some thoughts towards such a system are outlined.

The basic observation is that construction algorithms have more or less many *control parameters* by which their behavior can be influenced. These parameters are either offered to the user who may choose the best ones according to his feeling, or their values are determined in the program based on *features* of the current state of the algorithm. More flexibility can be achieved if both, features and control parameters, are made accessible outside the program. This allows to introduce a *control unit* which decides at each step of the algorithm at which a decision has to be taken, on the choice of all parameters dependent on the values of all features. The control unit can be the user or a control program.

The behavior of the control program is directed by *rules*. Changing of rules changes its behavior. The user may define rules that adapt the behavior of the algorithm to the type of data sets, or even to the type of a local subset of the data set.

The user may formulate these rules based on his experience. A more advanced approach is that the system learns the rules from the behavior of the user. This is achieved by logging the decisions taken by the user when he plays the role of the control unit. The logged decision patterns are generalized into decision rules.

If the user takes over the role of the control unit, he may be supported in his decision by the system, in that the system enumerates possible solutions. Since usually the number of solutions is too large to present them exhaustively, the user must have the possibility to control the process of enumeration. This can again be done by rules expressing constraints based on features of the solution. The rules can be refined on-line by the user.

It is a challenge of further research to develop mechanisms and systems along these ideas.

8. Acknowledgements

The authors would like to thank all researchers mentionend in this paper who have provided us generously with the original pictures and additional information. Prof. Georg Fleischmann and Oliver Bunsen, Academy of Media Arts in Cologne (Germany), provided us with the point set in Figure 18. Thomas Schreiber, Prof. Guido Brunnett and Frank Isselhard, University of Kaiserslautern, helped us by re-computing their reconstruction results and pictures especially for this paper. For implementations done in relation with this paper, the Delaunay triangulation and Alpha–Shape software as well as test data of Ernst Peter Mücke, University of Illinois at Urbana–Champaign, were used. Special thanks go to our colleague Frank Weller for valuable discussions.

References

1. Maria-Elena Algorri and Francis Schmitt. Surface reconstruction from unstructured 3d data. *Computer Graphics forum*, 15(1):47–60, 1996.

2. Dominique Attali. *r*-regular shape reconstruction from unorganized points. In *ACM Symposium on Computational Geometry*, pages 248–253, 1997. Nice, France.

3. Andreas Baader. *Ein Umwelterfassungssystem für multisensorielle Montageroboter.* PhD thesis, Universität der Bundeswehr, Munich, Germany, 1995. Fortschrittberichte, VDI Reihe 8 Nr. 486, ISBN 3-18-3 48608 - 3, ISSN 0178-9546, (in German).

4. Andreas Baader and Gerd Hirzinger. Three–dimensional surface reconstruction based on a self–organizing feature map. In *Proc. 6th Int. Conf. Advan. Robotics*, pages 273–278, 1993. Tokyo.

5. Andreas Baader and Gerd Hirzinger. A self–organizing algorithm for multisensory surface reconstruction. In *International Conf. on Robotics and Intelligent Systems IROS '94*, September 1994. Munich, Germany.

6. Chandrajit Bajaj, Fausto Bernardini, and Guoliang Xu. Reconstructing surfaces and functions on surfaces from unorganized 3d data. *Algorithmica*, 19:243–261, 1997.

7. Chandrajit Bajaj and D. Schikore. Error-bounded reduction of triangle meshes with multivariate data. In *Proceedings of SPIE Symposium on Visual Data Exploration and Analysis III*, pages 34–45, January 1996. SPIE.

8. Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. *Computer Graphics Proceedings, SIGGRAPH '95,Annual Conference Series*, pages 109–118, 1995.

9. Fausto Bernardini. *Automatic Reconstruction of CAD Models and Properties from Digital Scans*. PhD thesis, Department of Computer Science, Purdue University, 1996.

10. Fausto Bernardini and Chandrajit Bajaj. Sampling and reconstructing manifolds using alpha-shapes. In *Proc. of the Ninth Canadian Conference on Computational Geometry*, pages 193–198, August 1997. Also available as: Technical Report CSD-97-013, Department of Computer S ciences, Purdue University, 1997.

11. Fausto Bernardini, Chandrajit Bajaj, Jindong Chen, and Daniel R. Schikore. Automatic reconstruction of 3d cad models from digital scans. *Submitted for publication*, 1997. Also available as: Technical Report CSD-97-012, Department of Computer Sciences, Purdue University, 1997.

12. Eric Bittar, Nicolas Tsingos, and Marie-Paule Gascuel. Automatic reconstruction of unstructured data: Combining a medial axis and implicit surfaces. *Computer Graphics forum*, 14(3):457–468, 1995. Proceedings of EUROGRAPHICS '95.

13. Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, October 1984.

14. Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Computer Graphics Proceedings, SIGGRAPH '96, Annual conference series*, pages 303–312, 1996.

15. Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

16. Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley and Sons, Inc., 1973.

17. Matthias Eck and Hugues Hoppe. Automatic reconstruction of b–spline surfaces of arbitrary toplogical type. *Computer Graphics Proceedings, SIGGRAPH '96, Annual Conference Series*, pages 325 – 334, 1996.

18. Herbert Edelsbrunner. Weighted alpha shapes, 1992. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

19. Herbert Edelsbrunner and Ernst Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994. Also as Technical Report UIUCDCS-R-92-1734, Department of Computer Science, 1992, University of Illinois at Urbana-Champaign.

20. Eric Ferley, Marie-Paule Cani-Gascuel, and Dominique Attali. Skeletal reconstruction of branching shapes. *Computer Graphics forum*, 16(5):283–293, December 1997.

21. Pascal Fua and Peter T. Sander. From points to surfaces. In Baba C. Vemuri, editor, *Geometric Methods in Computer Vision*, pages 286–296. Proc. SPIE Vol. 1570, 1991.

22. Pascal Fua and Peter T. Sander. Reconstructing surfaces from unstructured 3D points. In *Proc. Image Understanding Workshop*, pages 615–625, San Diego, CA, 1992.

23. Pascal Fua and Peter T. Sander. Segmenting unstructured 3D points into surfaces. In G. Sandini, editor, *Computer Vision : ECCV '92, Proc. Second European Conference on Computer Vision*, pages 676–680, Santa Margherita Ligure, Italy, May 1992. Springer.

24. Baining Guo. Surface reconstruction: from points to splines. *Computer-Aided Design*, Vol. 29(No. 4):269–277, April 1997.

25. Baining Guo, Jai Menon, and Brian Willette. Surface reconstruction using alpha–shapes. *Computer Graphics forum*, Vol. 16(No. 4):177–190, October 1997.

26. Gabor T. Herman and Hsun Kao Liu. Three-dimensional displays of human organs from computed tomograms. *Computer Graphics and Image Processing*, 9:1–21, January 1979.

27. Hugues Hoppe. *Surface Reconstruction from Unorganized Points*. PhD thesis, Univ. of Washington, Seattle WA, 1994.

28. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, July 1992. Proceedings of Siggraph '92.

29. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *Computer Graphics Proceedings*, Annual Conference Series, pages 21–26, New York, 1993. ACM Siggraph. Proceedings of Siggraph '93.

30. Josef Hoschek and Dieter Lasser. *Fundamentals of Computer Aided Geometric Design*. A.K. Peters, 1993.

31. Frank Isselhard, Guido Brunnett and Thomas Schreiber. Polyhedral reconstruction of 3d objects by tetrahedra removal. Technical report, Fachbereich Informatik, University of Kaiserslautern, Germany, February 1997. Internal Report No. 288/97.

32. James V. Miller, David E. Breen, William E. Lorensen, Robert M. O'Bara, and Michael J. Wozny. Geometrically deformed models: A method for extracting closed geometric models from volume data. *Computer Graphics*, pages 217–226, July 1991. Proceedings of SIGGRAPH '91.

33. William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

34. Robert Mencl. A graph–based approach to surface reconstruction. *Computer Graphics forum*, 14(3):445–456, 1995. Proceedings of EUROGRAPHICS '95, Maastricht, The Netherlands, August 28 - September 1, 1995.

35. Robert Mencl and Heinrich Müller. Graph–based surface reconstruction using structures in scattered point sets. In *Proceedings of CGI '98 (Computer Graphics International), Hannover, Germany, June 22th–26th 1998. (to appear)*, 1998. similar version also available as Research Report No. 661, 1997, Fachbereich Informatik, Lehrstuhl VII, University of Dortmund, Germany.

36. Ernst Peter Mücke. *Shapes and implementations in three–dimensional geometry*. PhD thesis, Department of Computer Science, University of Illinois at Urbana–Champaign, September 1993.

37. Heinrich Müller and Arnold Klingert. Surface interpolation from cross sections. In H. Hagen, H. Müller, and G.M. Nielson, editors, *Focus on Scientific Visualization*, pages 139–189. Springer Verlag, 1993.

38. Shigeru Muraki. Volumetric shape description of range data using "blobby model". *Computer Graphics*, pages 217–226, July 1991. Proceedings of SIGGRAPH '91.

39. Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer Verlag, 1985.

40. Gerhard Roth and Eko Wibowoo. An efficient volumetric method for building closed triangular meshes from 3–d image and point data. In *Graphics Interface '97*, pages 173–180, 1997.

41. Detlef Ruprecht and Heinrich Müller. Spatial free form deformation with scattered data interpolation methods. *Computers and Graphics 19*, pages 63–71, 1995.

42. Thomas Schreiber. Approximation of 3d objects. In *Proceedings of the 3rd Conference on Geometric Modeling*, Dagstuhl, Germany, 1997. accepted for a supplementary issue of the journal *Computing* (Springer Verlag).

43. Thomas Schreiber and Guido Brunnett. Approximating 3d objects from measured points. In *Proceedings of 30th ISATA*, Florence, Italy, 1997.

44. Gilbert Soucy and Frank P. Ferrie. Surface recovery from range images using curvature and motion consistency. *Computer Vision and Image Understanding*, 65(1):1–18, 1997.

45. Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics*, 26:185–194, July 1992. Siggraph '92 Proceedings.

46. Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

47. Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *Computer Graphics Proceedings, SIGGRAPH '94, Annual conference series*, pages 311–318, 1994.

48. Remco C. Veltkamp. Closed object boundaries from scattered points. In *Lecture Notes in Computer Science 885*. Springer Verlag, 1994.

49. Remco C. Veltkamp. Boundaries through scattered points of unknown density. *Graphics Models and Image Processing*, 57(6):441–452, November 1995.

50. Frank Weller. Stability of voronoi neighborship under perturbations of the sites. In *Proceedings 9th Canadian Conference on Computational Geometry*, 1997. Kingston, Ontario, Canada, August 11–14.