University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

# Robust Surface Reconstruction Strategy for Large Clouds of Points

State of the Art and Concept of Doctoral Thesis

**Alexandr Jemĕljanov**

Technical Report No. DCSE/TR-2003-07
May, 2003

Distribution: public

Technical Report No. DCSE/TR-2003-07
May, 2003

# Robust Surface Reconstruction Strategy for Large Clouds of Points

**Alexandr Jeměljanov**

## Abstract

This work presents a strategy for creating a CAD model of an existing physical object from a scanned point cloud containing badly scanned regions. This strategy can be applied for processing big clouds. Formalization of properties of point clouds and surface reconstruction algorithms from the point of view of the given problem is made. This work describes solutions for several problems, which lie outside the field of usually considered surface reconstruction problems. Also, several algorithms designed within the framework of the strategy are described. Majority of them can be used for tasks of general-purpose surface reconstruction. The offered work contains also an overview of commonly used approaches of surface reconstruction.

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
30614 Pilsen
Czech Republic

## Contents

# 1. INTRODUCTION

## 1.1 Formulation of the problem

The problem of creating a CAD model for an existing physical object from a given point set of the object surface is important for many fields of science and industry. In most cases it is possible to receive the input point cloud having necessary properties (density, linear and angular isotropy). In this case a model can be successfully constructed by the application of existing algorithms. However, in some cases, obtaining of the initial point cloud with good enough characteristics can be complicated or impossible. Typical examples are architectural monuments, the objects explored from distance, e.g., objects in space. For the cloud of points obtained from these objects, there is a typical combination of regions having good allocation of points, and regions with unsatisfactory allocation of points, or without them.

## 1.2 State of the art

For general description of basic groups of existing surface reconstruction methods survey [MM97] is partially cited below.

### 1.21 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 [LC87]. Typical examples are regular grids, adaptive schemes like octrees, or irregular schemes like tetrahedral meshes. Many of them can also be applied to surface construction. The goal of construction algorithms based on spatial subdivision is to find cells related to the shape described by *P*. The selection of the cells can be done in roughly two ways: surface-oriented and volume-oriented.

#### 1.211 Surface-oriented cell selection

The surface-oriented approach consists of the following basic steps.
Surface-oriented cell selection:
1. Decompose the space in cells.
2. Find those cells that are traversed by the surface.

3. Calculate a surface from the selected cells.

**The approach of Algorri and Schmitt**

An example for surface-oriented cell selection is the algorithm of Algorri and Schmitt [AS96] (figure 1.211-1). 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 outer quadrilaterals of the selected voxels are taken as a first approximation of the surface. This resembles the cuberille approach of volume visualization [HL79].

In order to get a more 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 1.232.

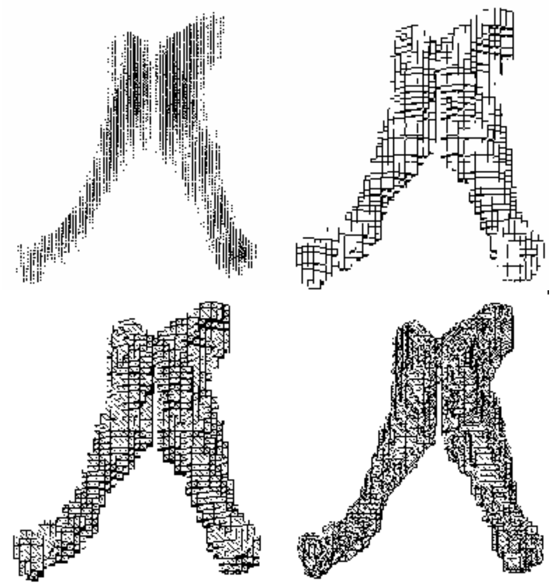


Figure 1.211-1

**The approach of Hoppe et al.**

Another possibility of surface-oriented cell selection is based on the distance function approach of Hoppe [HDD92, HDD93, HH94] (figure 1.211-2). 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 1.221.

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

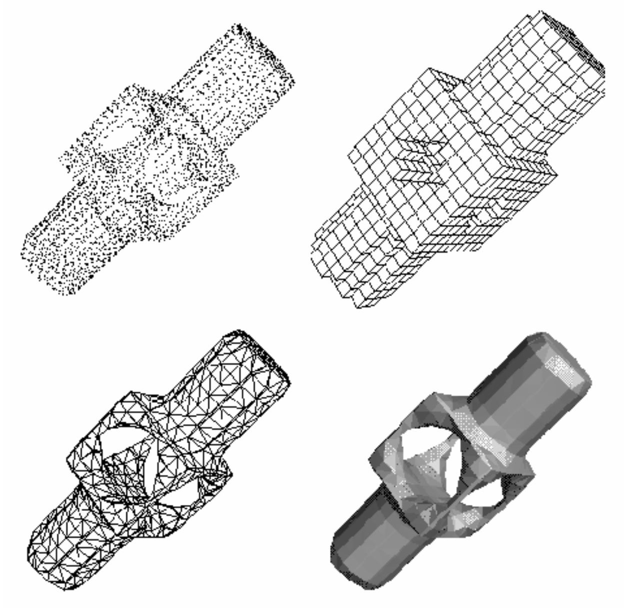


Figure 1.211-2

A similar algorithm was suggested by Roth and Wibowoo [RGW97]. It differs from the approach of Hoppe et al. in the way the distance function is calculated, cf. section 1.221. Furthermore, the special cases of profile lines and multiple view range data are considered besides scattered point data.

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

### The approach of Bajaj, Bernardini et al.

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

The algorithm also requires a signed distance function. For this purpose, a first approximate surface is calculated in a preprocessing phase. The distance to this surface is used as distance function. The approximate surface is calculated using $\alpha$-solids which will be explained in section 1.212.

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-Bezier 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 [PS85].

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

In Bernardini et al. [BBC97] an extension and modification of this algorithm is formulated [BBX97, BB97]. The algorithm consists of an additional mesh simplification step to reduce the complexity of the mesh represented by the $\alpha$-solid [BS96]. The reduced mesh is used in the last step of the algorithm for polynomial-patch data fitting using Bernstein-Bezier patches for each triangle (by interpolating the vertices and normals and by approximating data points in its neighborhood). Additionally, the representation of sharp features can be achieved in the resulting surface.

**Edelsbrunner's and Mucke's alpha-shapes**

Edelsbrunner and Mucke [EPM92, EPM93, HE92] 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 [PS85]. 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 *$\alpha$-balls* as eraser sphere with radius

$\alpha$. 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 *$\alpha$-shape* is a collection of points, edges, faces, and tetrahedra.

In the third step, the triangles are extracted out of the $\alpha$-shape which belong to the desired surface, using the following rule. Consider the two possible spheres of radius $\alpha$ through all three points of a triangle of the $\alpha$-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 $\alpha$. Since $\alpha$ 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 1.211-3. If $\alpha$ is too small, gaps in the surface can occur, or the surface may become fragmented.

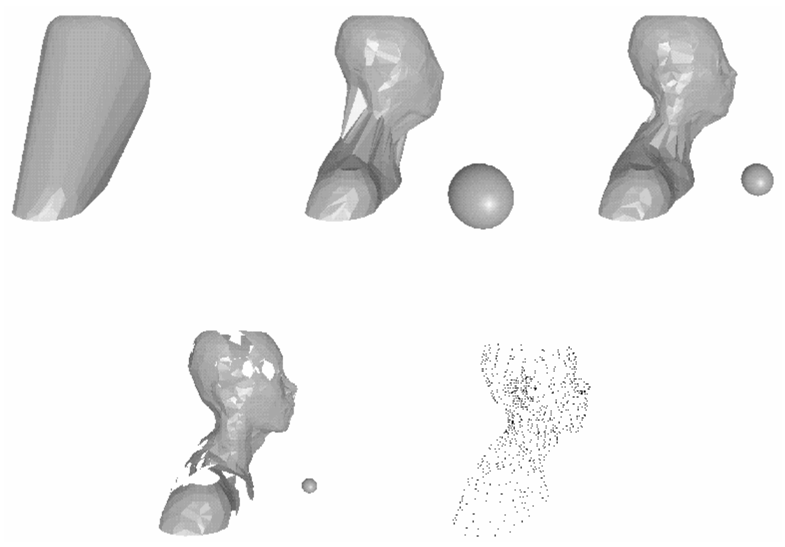


Figure 1.211-3

Guo et al. [GMW97] also make use of $\alpha$-shapes for surface reconstruction but they propose a so-called *visibility* algorithm for extracting those triangles out of the $\alpha$-shape which represent the simplicial surface.

## Attali's normalized meshes

In the approach of Attali [DA97], 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.

## 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 $d(p_i p_i^{'}) \leq \varepsilon$ holds for all $p_i \in P$, $p_i^{'} \in P'$, $i = 1,...,n$. 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*(Voronoi neighbors) [FW97].

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 [DA97], cf. section 1.211. Further research is necessary in order to make stability useful for surface construction.

## 1.212 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 [JB84, IBS97, RV94]. Another methodology is the use of the Voronoi diagram to describe the constructed solid by a “skeleton” [SB97, DA97].

### 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 1.212-1.

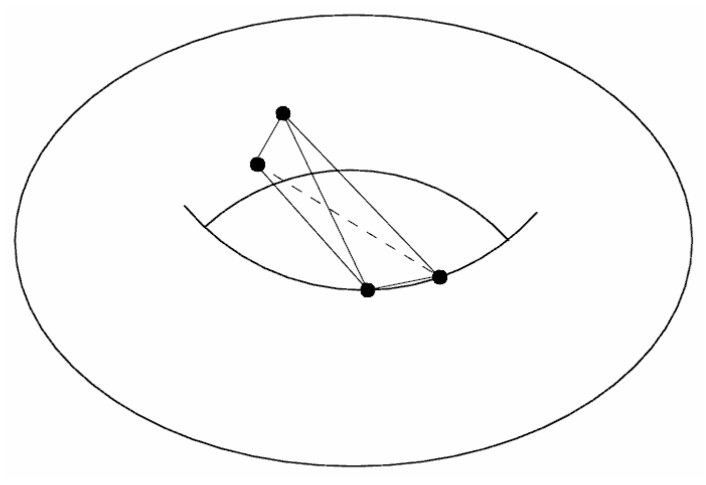


Figure 1.212-1

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.

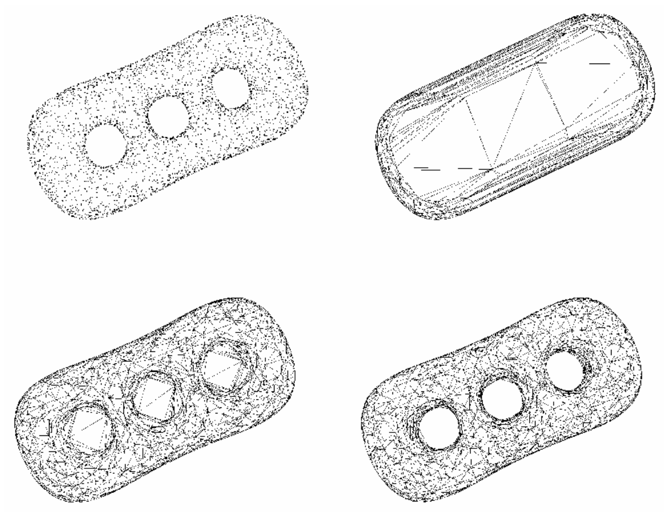


Figure 1.212-2

**The approach of Isselhard, Brunnett, and Schreiber**

The approach of [IBS97] is an improvement of the volume-oriented algorithm of Boissonnat [JB84]. While Boissonnat cannot handle objects with holes, the deletion procedure of this approach is modified so that construction of holes becomes possible.

As before, the algorithm starts with the Delaunay triangulation of the point set. An incremental tetrahedron 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. 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 [JB84].

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 1.212-2.

**The $\gamma$-indicator approach of Veltkamp**

To describe the method of Veltkamp [RV94, RV95] some terminology is required. A *$\gamma$-indicator* is a value associated to a sphere through three boundary points of a polyhedron which is positive or negative, cf. figure 1.212-3 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 $\gamma$-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.

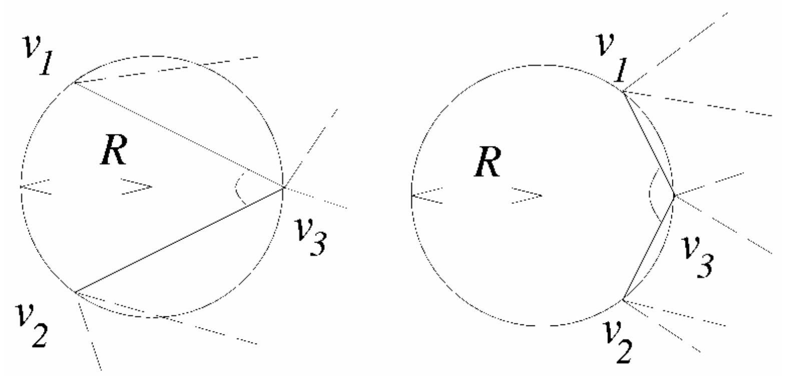


Figure 1.212-3

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 [JB84]. The tetrahedron with the largest γ-indicator value is removed and the boundary is updated. This process continues until all points lie on the boundary, or 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 1.212-4.

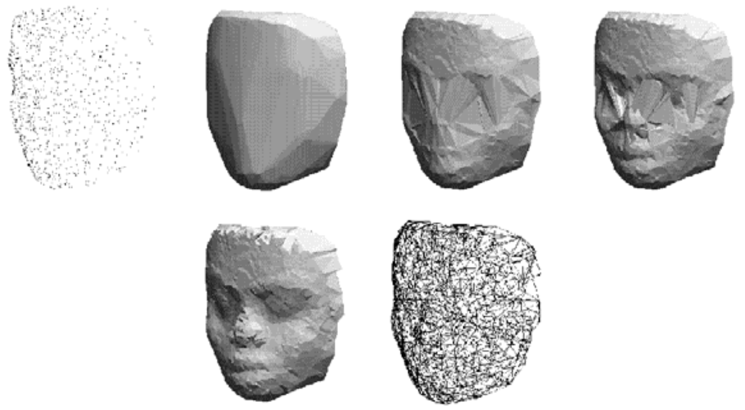


Figure 1.212-4

## The approach of Schreiber and Brunnett

The approach of Schreiber and Brunnett [TS97, SB97] uses properties of the Voronoi diagram of the given point set for tetrahedra removal. The *Voronoi diagram* of a point set *P* is a partition of the space in regions of nearest neighborhood. For each point *p* in *P*, it contains the region of all points in space that are closer to *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 1.212-5. 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.

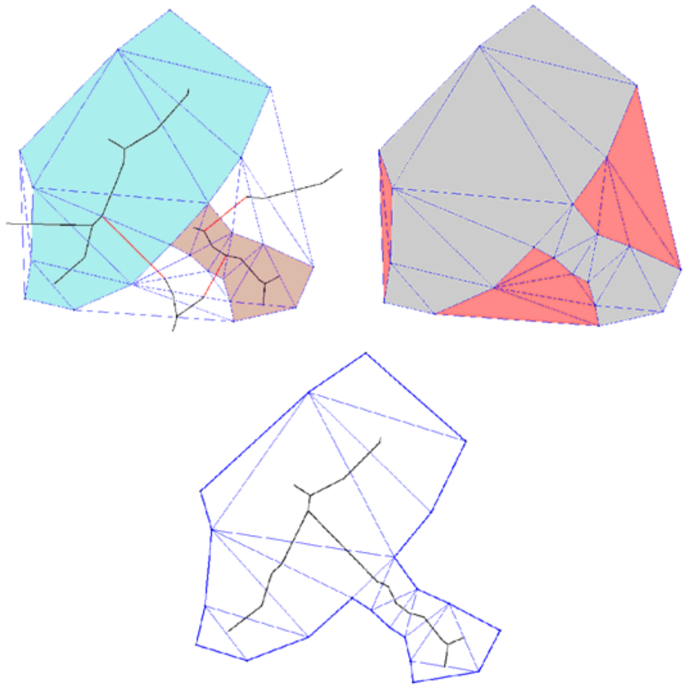


Figure 1.212-5

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

Two pruning rules have been developed for that purpose:

1. All those edges 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. In order to find the inside regions, we add the complement of the convex hull as further region to the set of subtree regions. The algorithm starts with a point on the convex hull which is incident to exactly two regions. The region different from the complement of the convex hull is classified “inside”. Then the label “inside” is propagated to neighboring regions by again considering points that are incident to exactly two regions.

After all regions have been classified correctly, the boundary of the constructed shape is obtained as the boundary of the union of the region labeled “inside”.

An adaption of this method to three dimensions is possible.

**The $\alpha$-solids of Bajaj, Bernardini et al.**

Bajaj, Bernardini et al. [BBX95, BBX97, BB97, BBC97] calculate so-called *$\alpha$-solids*. While $\alpha$-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 [JB84]. To overcome the approximation problems inherent to $\alpha$-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 $\alpha$-shape approach has correctly recognized the larger structures of the object.

## 1.22 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 section 1.21 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 [BBX95].

Besides marching cubes construction of surfaces as explained in section 1.211, 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 1.222.

### 1.221 Calculation of distance functions

#### The approach of Hoppe et al.

Hoppe et al. [HDD92, HH94] suggest the following approach. At the beginning, for each point $p_i$ an estimated tangent plane is computed. The tangent plane is obtained by fitting the best approximating plane in the least square sense [DH73] into a certain number *k* of points in the neighborhood of $p_i$. 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* (figure 1.221-1). 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 $o_i, o_j$ 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.

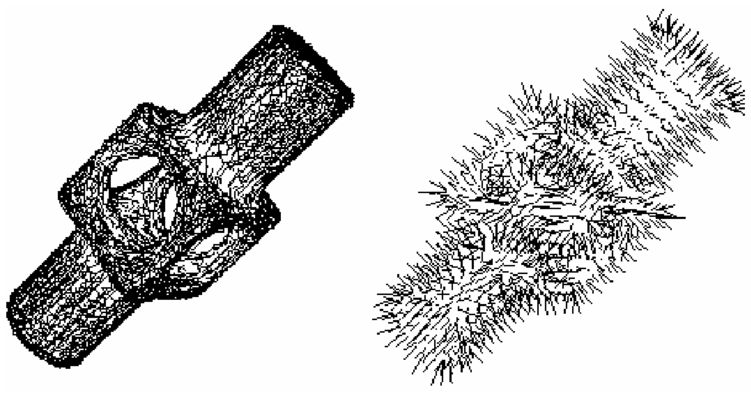


Figure 1.221-1

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.

**The approach of Roth and Wibowoo to distance functions**

The goal of the algorithm of Roth and Wibowoo [RGW97] 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°.

### 1.222 Bittar's et al. surface construction by medial axes

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

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

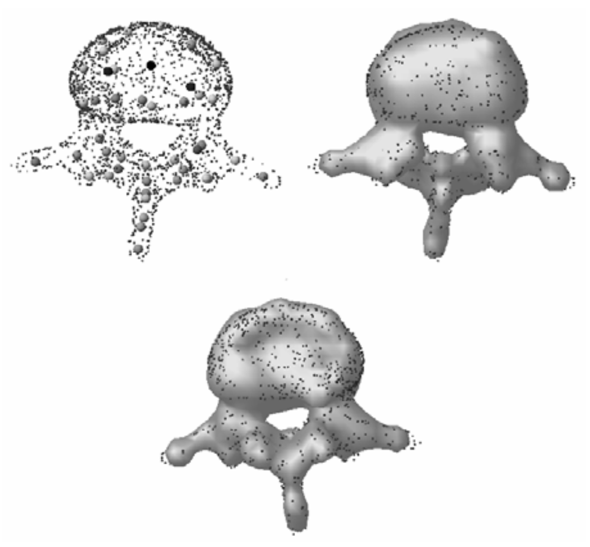


Figure 1.222-1

The desired surface is calculated by distributing centers of spheres on the medial, cf. figure 1.222-1. 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 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.

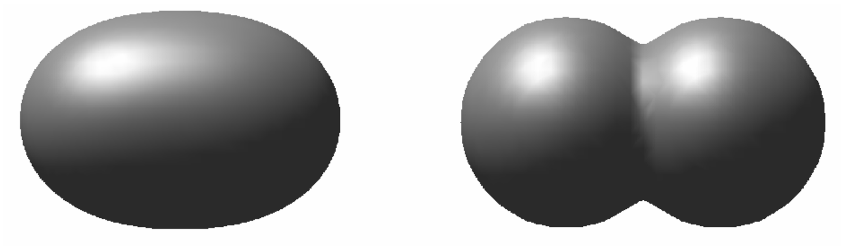


Figure 1.222-2

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 1.222-2. Also the connectness of the resulting solid can be influenced by the shape function cf. figure 1.222-3.

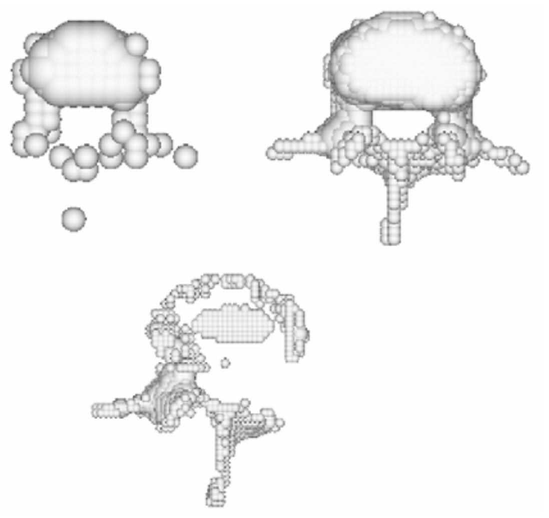


Figure 1.222-3

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.

A result of this algorithm is shown in figure 1.222-1.

## 1.23 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 [SM91]. Terzopoulos, Witkin and Kass [TM91, TWK88] made use of *deformable superquadrics* which have to fit the input data points.

Miller et al. [MBL91] 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 similar 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 [ST92]. Each particle owns several parameters which are updated during the modeling simulation. By modeling the interaction between the particles themselves the surface is being modeled using forces and repulsion. As an extension Szeliski and Tonnesen describe how their algorithm can be extended for automatic 3D reconstruction. At each sample location one particle with appropriate parameters is generated. The gaps between the sample points (particles, respectively) are filled by growing particles away from isolated points and edges. After having a rough approximation of the current surface the other particles are rejusted to smooth the surface.

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

### 1.231 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. [HJL93]. 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 [RM95].

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

### 1.232 The approach of Algorri and Schmitt

The idea of Algorri and Schmitt [AS96] is to translate given approximate triangular mesh into a physical model, cf. figure 1.232-1. 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.

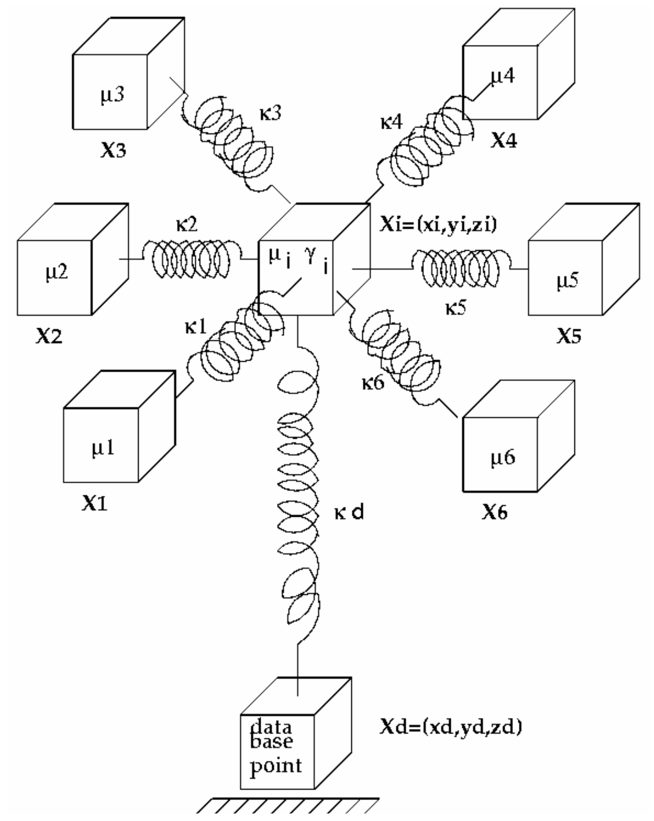
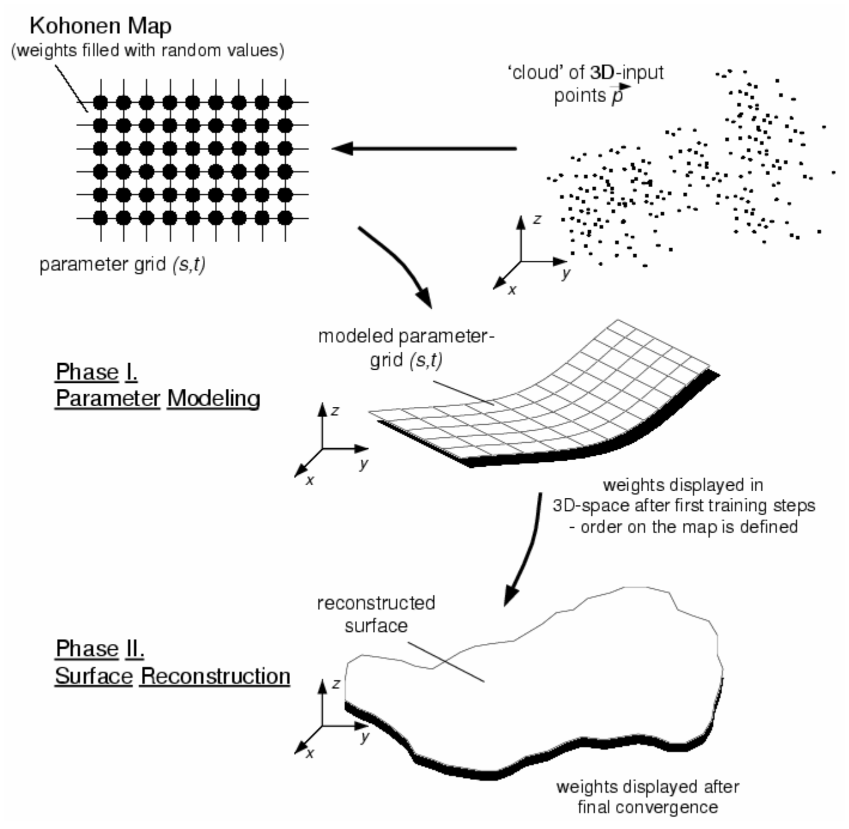


Figure 1.232-1

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 1.211.



Kohonen Map
(weights filled with random values)
'cloud' of 3D-input points $\vec{p}$
parameter grid (s,t)
modeled parameter-grid (s,t)
Phase I.
Parameter Modeling
weights displayed in 3D-space after first training steps - order on the map is defined
reconstructed surface
Phase II.
Surface Reconstruction
weights displayed after final convergence
z
y
x

Figure 1.233-1

### 1.233 Kohonen feature map approach of Baader and Hirzinger

The Kohonen feature map approach of Baader and Hirzinger [BH93, BH94] can be seen as another implementation of the idea of surface construction by warping. Kohonen's feature map is a two-dimensional array of units (neurons), cf. figure 1.233-1. Each unit $u_j$ has a corresponding weight vector $\overrightarrow{w_j}$ . 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 $o_j = \overrightarrow{w_j}\vec{i}$ . 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 $\overrightarrow{w_j}(t+1) = \overrightarrow{w_j}(t) + \varepsilon_i(\vec{i} - \overrightarrow{w_j}(t))$

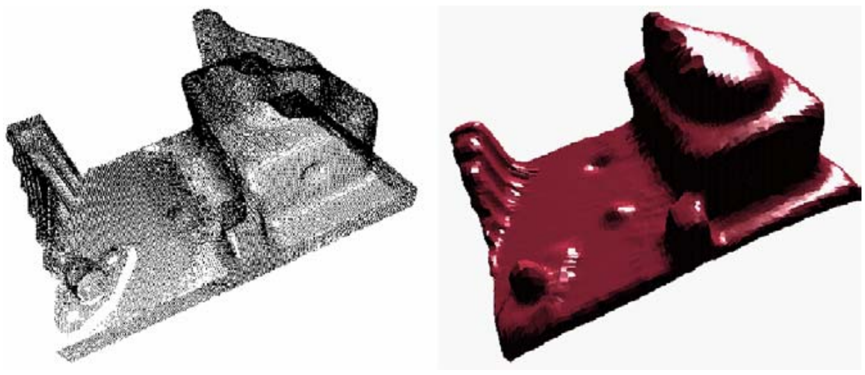


Figure 1.233-2

Note that after this update the weight vectors have to be normalized again. The value $\varepsilon_j = \eta h_j$ contains two values, the learning rate $\eta$ 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 1.233-2.

## 1.24 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.

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 Muller described in section 1.242.

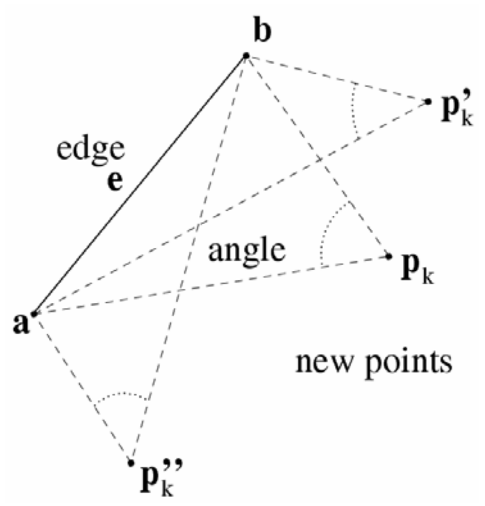


Figure 1.241-1

### 1.241 Boissonat's surface-oriented approach

Boissonnat's surface oriented contouring algorithm [JB84] 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 new triangle, that is, the point sees edge boundary edge under the maximum angle, cf. figure 1.241-1. The algorithm terminates if there is no free edge available any more. The behavior of this algorithm can be seen in figure 1.241-2.

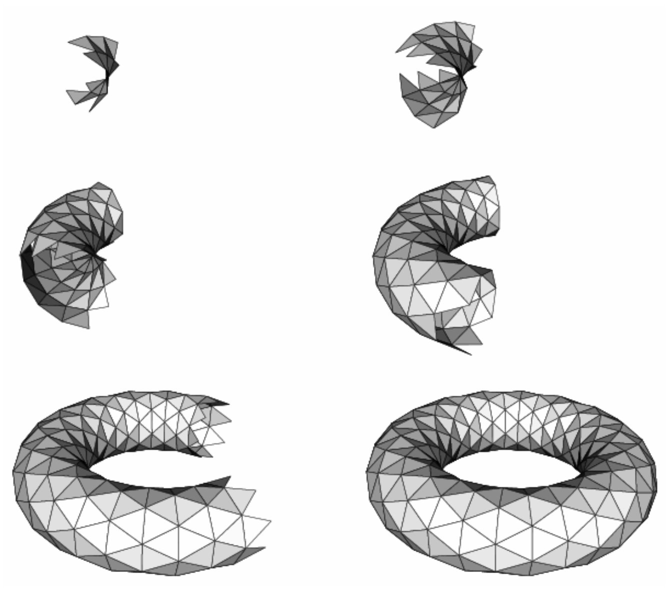


Figure 1.241-2

### 1.242 Approach of Mencl and Muller

The solution of Mencl and Muller consists of seven main steps [RM95, MM97]:

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 1.242-1. 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 Muller 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. Finally, the triangles are filled into this surface description graph by using a rule system to assure a resulting surface with high accuracy.

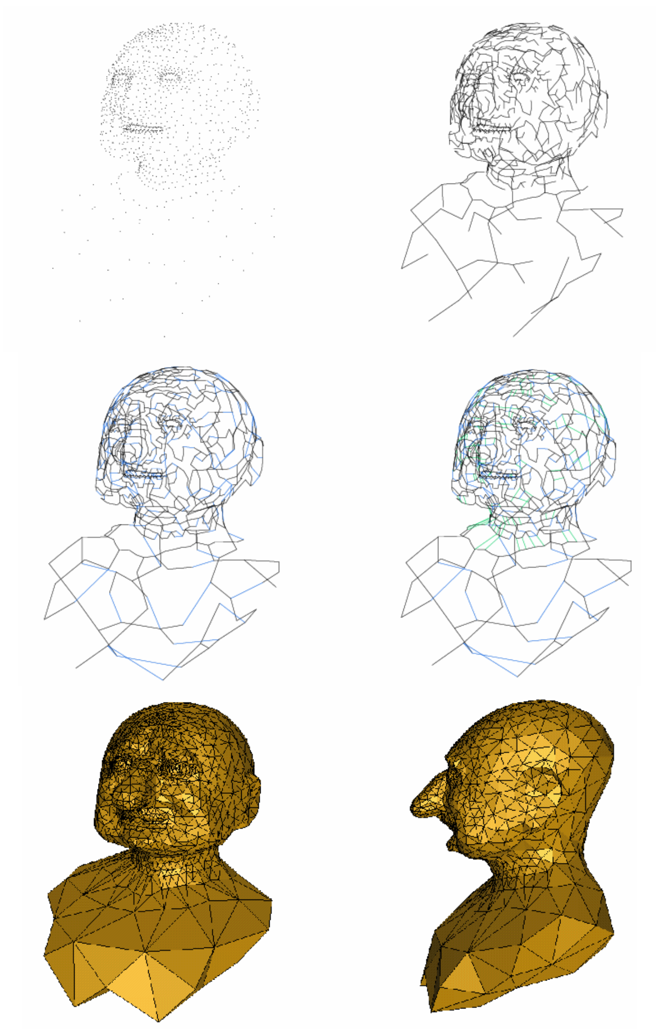


Figure 1.242-1

As a main concept, Mencl and Muller introduce the concept of feature recognition and clustering to improve the accuracy of the surface description graph according to the assumed surface of the object [MM97]. 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 Mobius strips, for example.

### 1.3 A motive for further researches

Generally, for our task no one of the existing algorithms is enough suitable. The algorithms with a low cost aren't enough robust to reconstruct damaged point clouds, and the robust algorithms are too costly for processing big point clouds.

## 2. A GENERAL SURFACE RECONSTRUCTION STRATEGY

### 2.1 Formalization of properties of a reconstructed surface

**Definition 2.1-1.** A process of surface reconstruction from a point cloud is considered as successful if as a result we have a closed surface bounding a coherent volume and containing all the points of the cloud.

A result of an unsuccessful surface reconstruction can be presented as the aggregate of correctly reconstructed surface regions and points, which don't belong to such regions (let's call these points *free points*). A boundary of a correctly reconstructed region is always closed. This boundary can be the external boundary of the given isolated reconstructed region (let's call such region *island*) or the boundary of a hole in this region.

For each boundary we need to determine its status (is it the boundary of an island or the boundary of a hole). Let's consider, that the boundary is a smooth curve and the surface region bounded by this boundary is a smooth surface region, because any real reconstructed surface region (its boundary) can by approximated by a smooth surface (curve) as accurate, as it is needed. For the boundary let's determine a *main chord vector* as a vector connecting two points on the boundary with condition, that these points split the boundary into two parts with equal length. Let we have a main chord vector (vector $\overrightarrow{AB}$ in figure 2.1-1). In an enough close neighborhood of point *B*, the boundary line can be approximated by the tangent line in this point. In the same neighborhood the reconstructed surface can be approximated by the tangent plane in *B*. The tangent line lies in the tangent plane and splits it into two half-planes. Let's call the half-plane that approximating the reconstructed surface region the *internal half-plane* in *B*. Let's define the angle $\phi$ in *B* as the angle between $\overrightarrow{AB}$ and the internal half-plane in *B*. For a boundary (*L*) let's determine the *boundary integral value* (*D*):

$$D = \oint_L \cos\phi(l)dl \quad (2.1\text{-}1).$$

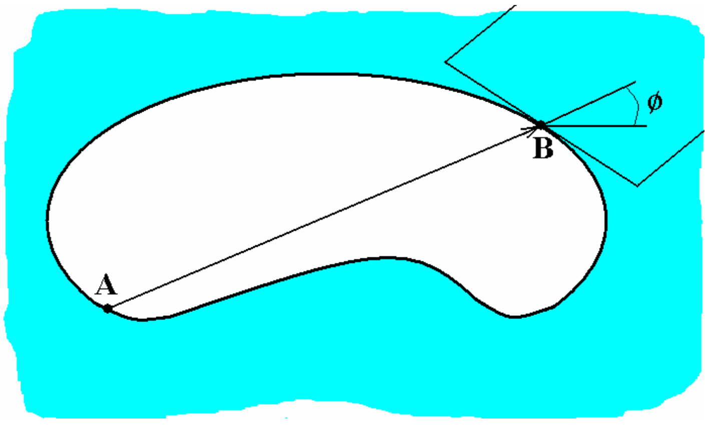


Figure 2.1-1

**Definition 2.1-2**. A boundary of a reconstructed surface is considered as the boundary of a hole, if for this boundary $D > 0$ or the boundary of an island otherwise.

**Definition 2.1-3.** Let's call the average plane of all the points of the boundary of a hole the *own plane* of the hole (the boundary).

**Definition 2.1-4.** If the boundary of a hole has unambiguous projection onto its own plane, then the hole is considered as a *simple hole* or as a *complex hole* otherwise.

**Definition 2.1-4.** A result of an unsuccessful surface reconstruction can be related to one of the next classes:

*CS1*: a coherent surface with simple holes inside;

*CS2*: a coherent surface with simple and complex holes inside;

*CS3*: the aggregate of several isolated islands with possible holes inside the islands.

## 2.2 Formalization of properties of point clouds and algorithms

Generally, the quality of a point cloud is a complex concept; it can't be expressed by a single scalar value or a low-dimensional set of such values. In addition, the quality of a point cloud makes sense only concerning a given surface reconstruction algorithm. At the present time many such algorithms having different properties and possibilities are designed.

**Definition 2.2-1.** An input point cloud contains information about the corresponding original surface as the coordinates of some subset of the surface points. Let's suppose that there is an algorithm that makes surface reconstruction only on the base of analysis of distances between points of the cloud. During surface reconstruction this algorithm don't make any assumptions concerning the original surface behavior and don't make any analyses of surface behavior in already reconstructed regions. Let's call such hypothetical algorithm *distance-analyzing algorithm* (*DAA*).

Let's appraise the quality of an input point cloud from the point of view of surface reconstruction by DAA. We need to estimate the next two things:

- possibility of a successful surface reconstruction in principle;
- cost of such reconstruction.

**Definition 2.2-2.** Later on, the notion of the distance between two points (*A* and *B*) of the surface of an object along this surface will be used. Let's define this distance as the length of a trajectory between the points on the surface. The trajectory is defined as intersection the surface and a plane containing the given points. The plane should minimize the integral:

$$\int_0^L (\phi(l))^2 \, dl \quad (2.2\text{-}1),$$

where

*L* is the length of a trajectory between *A* and *B*;

$\phi$ is the angle between external surface normal in a point on the trajectory and the plane.

Let's denote the distance between two points *A* and *B* along theirs original surface *{A,B}*.

For the beginning, let's consider the task of formalization of the quality of a point cloud in a neighborhood of a given point (*A*) of this cloud.

Among the factors having influence on the cost of surface reconstruction, the uniformity of sampled points is the most important. For *A* let's determine a set (*{B}*) of neighbors of 1-st order. Let *{B}* consists of *n* elements. The elements of *{B}* are *n* the closest (along the original surface) neighbors of *A* with condition, that the angle between any two vectors from *A* to these neighbors isn't smaller than $\alpha$. As *n* we choose the closest integer number to the averaged number of point's neighbors in triangular meshes - 6. As $\alpha$ we choose a typical restriction for the minimal angle in a triangulation – 30°.

**Definition 2.2-3.** Irregularity of points in a neighborhood of a given point *A* can be evaluated by the *local irregularity factor* (*u*) defined by the formula:

$$u = \frac{|A, B_{\max}|}{|A, B_{\min}|},$$

where $B_{\max}, B_{\min}$ – are the most remote and the closest points of *{B}* of *A*.

Let's define for each point of *{B}* (*B*) a 2-nd order neighbor of *A*. We define this point as the closest neighbor of *B* that does not belong to *{B}*. The set of all the 2-nd order neighbors of *A* we denote *{C}*.

**Definition 2.2-4.** Let's name a given point (*A*) of a point cloud a *critical point*, if for some point (*B*) of *{B}* of *A* and the corresponding point (*C*) of *{C}* of *A* the next inequalities are valid:

$$\{A,B\} < \{A,C\},$$
$$|A,B| > |A,C|.$$

If a given point is a critical point, then a DAA can't reconstruct the original surface correctly in a neighborhood of the point. Thus, presence of critical points is a criterion of possibility of surface reconstruction by a DAA in principle.

In the further consideration we will consider, that a point cloud hasn't any critical points. We estimate the quality of such cloud from the point of view of the cost of surface reconstruction by a DAA. A DAA we estimate by its possibility of successful surface reconstruction from a point cloud with a given quality. This method of estimation is enough universal and informative because:

1) a DAA-component is the basic for the majority of existing surface reconstruction algorithms;
2) practically used point clouds mainly have only a little share of critical points or haven't them.

**Definition 2.2-5.** With due regard for mentioned above, the quality of a point cloud is estimated by the two factors of irregularity - $U_m$ and $U_a$. They are the maximum and the average value of *u* in the point cloud correspondingly.

**Definition 2.2-6.** A given triangulation algorithm is estimated by two factors. They are the *robustness* (*R*) and the *speed* (*S*). *R* is the upper limit of $U_m$ of the point clouds, which can be successfully processed by the algorithm. *S* can be expressed by the formula: $S = N/T$, where *N* is the number of points of a point cloud; *T* is the time of surface reconstruction. *S* usually depends on $U_a$. The *cost* (*C*) of an algorithm is defined as the inverse value of *S*: $C = 1/S$.

As a whole, we have the next correlation between *C* and *R* for practically used surface reconstruction algorithms (for $U_a = const$):

$$C \sim kR^{\gamma} \quad (2.2\text{-}2),$$

where *k* is a constant factor, $\gamma$ has a value from the diapason 1.5 – 2.5.

### 2.3 Classification of point clouds and strategy S1

Let's consider the distribution of *u* inside a given point cloud. A typical distribution of this parameter for a little model scanned in a laboratory is shown in figure 2.3-1. For such case $U_a$ is insignificant, and $U_m$ does not differ considerable from $U_a$. Let's name clouds with such property *clouds of class CP1*. Surface reconstruction from a point cloud of the given class can be made by an algorithm having the appropriate $R = U_m$ with the next cost:

$$C = kR^{\gamma}N = kU^{\gamma}N \quad (2.3\text{-}1).$$

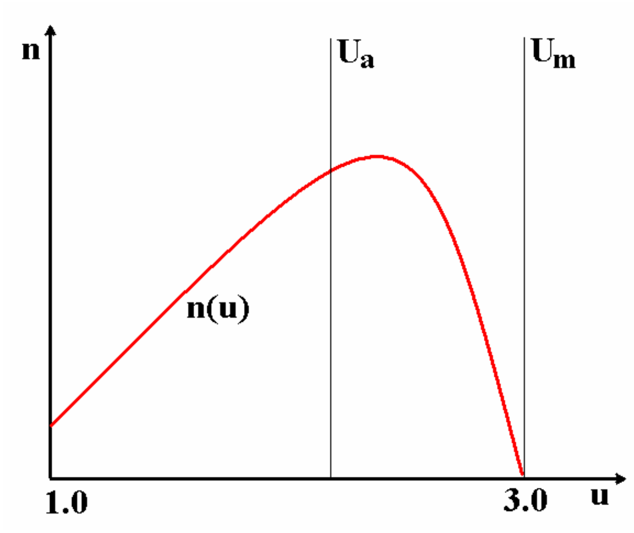


Figure 2.3-1

In figure 2.3-2 a typical distribution of *u* for another kind of point clouds is shown. Such clouds are obtained by physical scanning architectural objects and objects explored from distance. For these point clouds $U_a$ is still a relative little, but $U_m$ is essentially greater than $U_a$. It is a typical situation for the case of a good or average sampled surface with defects of the sampling. Such defects can be caused by physical defects of the surface, shielding, and so on. Let's name clouds having such property *clouds of class CP2*. In this case a surface reconstruction by a single algorithm is extremely costly due to power dependence in formula 2.3-1.

**Definition 2.3-1.** Let's pick out all the point clouds, which can be correctly processed by an algorithm having typical values of *R* and *C*. For the present we choose as such algorithm the algorithm described in [MV01]. These clouds we refer to class CS1. Let's pick out all the remaining point clouds, from which the typical algorithm can reconstruct at least 50% of the area of the corresponding sampled surfaces. These clouds we refer to class CS2. The remaining point clouds we consider practically unusable.

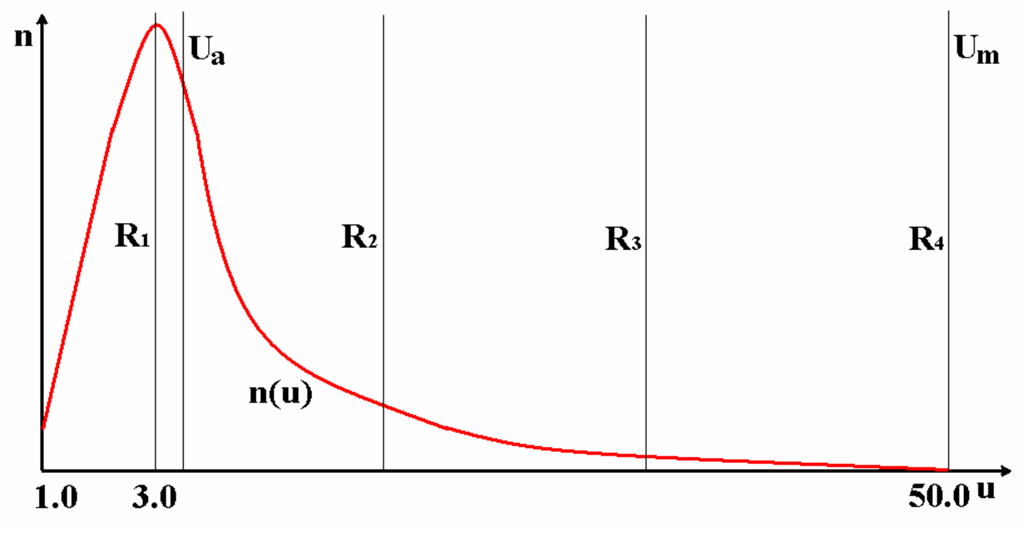


Figure 2.3-2

For a point cloud of class CP2 let's consider consecutive application of *q* algorithms having increasing values of robustness $R_i$; $i = \overline{1, q}$; $R_i < R_{i+1}$; $R_q = U_m$. In figure 2.3-2 a situation for $q = 4$ is shown. For the beginning the fastest (and having the least robustness) algorithm is applied. Let's call it *beginning algorithm.* This algorithm has deal with all the input points. After application of the beginning algorithm a stage of filtering is applied. At this stage we determine the set of points of correctly triangulated regions and the set of free points. Then the next (more robust and slow) algorithm is applied. This algorithm has deal with the current set of free points and boundaries of the correctly triangulated regions. And so on. The cost of surface reconstruction in the given case can be expressed by the formula:

$$C = \sum_{i=1}^{q} (kR_i^{\lambda} + F) \int_{R_{i-1}}^{Rq} n(u)du \quad \text{(2.3-2)},$$

where

$R_0 = 1.0$;

the integral expresses the number of input points for *i*-th algorithm;

*n(u)* is a function of points distribution concerning *u*;

*F* is the cost of the filtering stage per one point.

An algorithm of the described above strategy (let's call it *strategy S1*) is shown in figure 2.3-3.

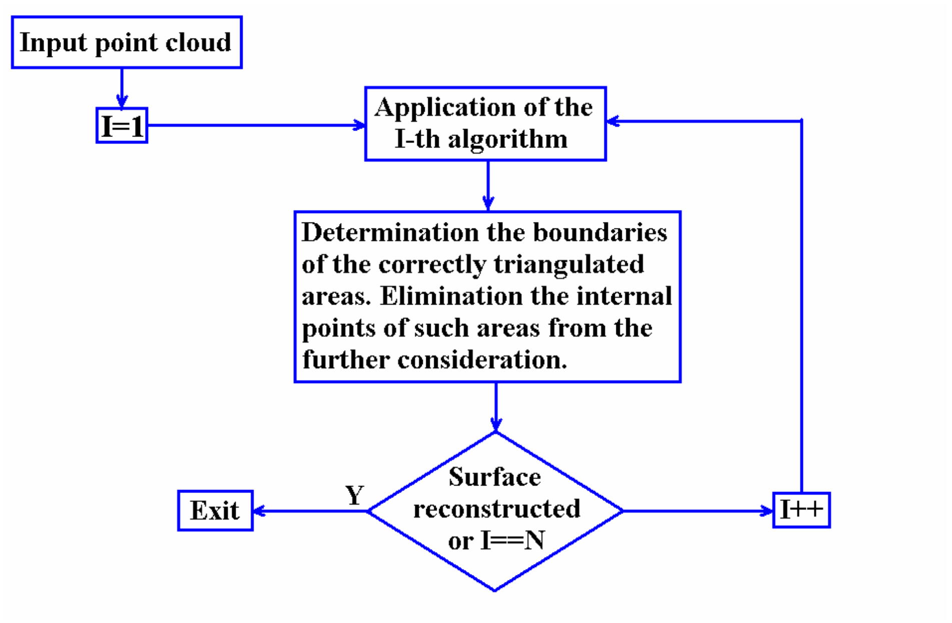

Input point cloud
I=1
Application of the I-th algorithm
Determination the boundaries of the correctly triangulated areas. Elimination the internal points of such areas from the further consideration.
Surface reconstructed or I==N
Y
Exit
I++

Figure 2.3-3

## 2.4 Strategy S2

Let’s consider an advanced strategy, which can reduce the total cost of surface reconstruction even more. This strategy consists of three main stages.

The first stage (*A*) is carried out according the strategy S1. As a result of this stage we obtain a partially reconstructed surface (let’s call it *A-surface*), which can be referred to one of the classes defined in (2.1-4).

**Definition 2.4-1.** Regions of an input point cloud correctly reconstructed at stage A we name *sufficiently sampled regions* (*SSR*).

After stage A the second stage (*B*) begins. The goal of this stage is reduction of the obtained A-surface from class CS3 to class CS2 and then to class CS1. Naturally, if the A-surface has class CS2 or CS1, then the first sub-stage of stage B or whole the stage is skipped. The basic idea of stage B is that for reduction of an A-surface to class CS1 it is enough to make a surface reconstruction only in several selected little regions. Therefore, in spite of the fact that for this reconstruction it is necessary to use a robust (and costly) algorithm, the total cost of this stage can be made a relative small.

After reduction the reconstructed surface to class CS1 the third stage (*C*) begins. At this stage surface reconstruction in remaining non-reconstructed regions is made. The total area of such regions is not considerable smaller than the total area of non-reconstruction regions obtained after application stage A. However, now these regions are represented by a set of simple holes, and for surface reconstruction inside them we don’t need to use costly algorithms.

So, during surface reconstruction in accordance of the described above strategy (let's call it *strategy S2*) the majority of input points are processed at stages A and C by enough simple and fast algorithms.

## 3. STAGE A

Within the framework of the given stage a beginning algorithm and an algorithm of the filtering stage were designed. The current implementation of the stage consists of subsequent application of these algorithms. In case of need the stage can be easily supplemented by subsequent application of a more robust algorithm(s), for example, [MV01].

### 3.1 A beginning algorithm

Generally, this algorithm uses the ideas of *greedy triangulation* (*GT*) and belongs to the group of interpolating methods. The GT of a point set in the plane is the triangulation obtained by starting with the empty set and at each step adding the shortest compatible edge between two of the points [BE95]. In 2D a compatible edge is defined to be an edge that crosses none of the previously added edges. This algorithm is an extension of 2D GT-strategy for 3D. It makes this algorithm to be very close to DAA.

For using GT-strategy in 3D a special very simple and fast test was designed. This test analyzes a topology of a created mesh at the place of prospective inclusion of an edge, and can be formulated as follows: if insertion of the current tested edge leads to an appearance of the edges having more than two adjacent triangles or leads to appearance of a tetrahedron, the edge is considered as incompatible and compatible otherwise. The given test does not use any floating-point operations, and is passed fast.

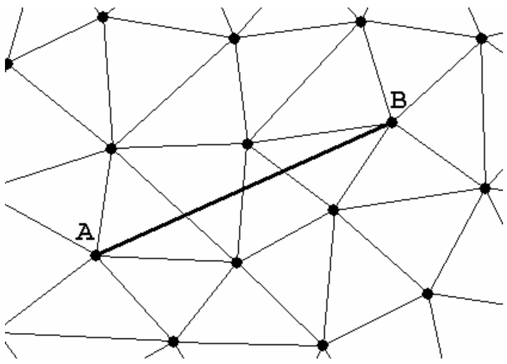


Figure 3.1-1

This test can't prevent appearance of edges having less than 2 adjacent triangles (edge *AB* in figure 3.1-1). Therefore, at the end of the algorithm we eliminate all such edges. Naturally, process based on such simple test can't provide 100% reliability for triangulation, but it is not necessary for a beginning algorithm. Typically we have about 95% correctly connected points in regions having $U_m \leq 6$.

## 3.2 Filtering

The sub-stage of filtering is intended to control of a result of application of a surface reconstruction algorithm applied before. On the input of this sub-stage there is some set of edges and triangles. On the output we have some set of correctly reconstructed regions and some set of free points. Free points haven't edges.

Generally, the sub-stage detects and marks points of the correctly triangulated regions. Simultaneously the sub-stage removes the edges and triangles, which don't create a correctly triangulation. The sub-stage can improve some errors of the applied before surface reconstruction algorithm, which are caused by generation of redundant edges and triangles. The sub-stage doesn't make generation of new edges and triangles.

**Condition 3.2-1.** On an input set of edges is imposed the next condition: each edge can have only one or two adjacent triangles.

For this sub-stage we use a variant of the well-known "umbrella" filtering [AGJ02]. Each point on the input can have one or several chains of adjacent triangles. These chains can be closed or open. Because of condition 3.2-1 such chains can't have any intersections (shared edges). Possible cases are shown in figure 3.2-1.

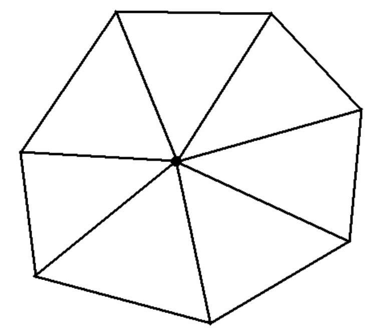


Figure 3.2-1a

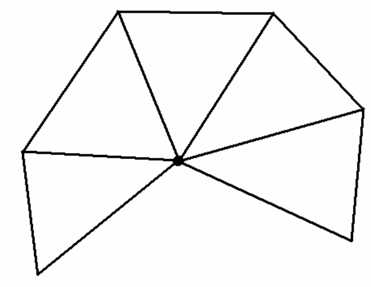


Figure 3.2-1b

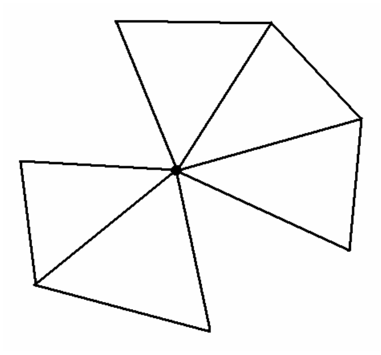


Figure 3.2-1c

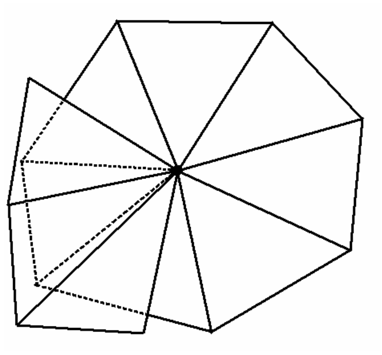


Figure 3.2-1d

**Lemma 3.2-1.** Can be easily proved, that a point of a correctly triangulated region can have only one chain of triangles. This chain is closed for an internal point (let's denote such point *ICP*), and is open for a boundary point (let's denote such point *BCP*).

The condition of lemma 3.2-1 is necessary, but is sufficient only if a given triangulated region is enough extensive. For example, vertices of a single isolated wrong triangle satisfy to this condition.

**Definition 3.2-1.** Let we have a triangulated region consist of points, which satisfy to the condition of lemma 3.2-1. We consider such region correctly triangulated, if each boundary point is connected at least with one internal point.

During the filtering each point can have one of the listed below statuses:
The general beginning status:
NT – the point is not tested.
The statuses after the umbrella-test application:
IC – the point has only one closed chain of triangles (figure 3.2-1a);
BC – the point has only one open chain of triangles (figure 3.2-1b);
MC – the point has more than one chain of triangles (figure 3.2-1c, d);
FR – the point hasn't any edges and triangles.
The status for more effective organization of the filtering process:
CF – the point belongs to a current front of corrected points.

In addition, if a triangle has at least one IC or BC point as a vertex, then this triangle is marked as an *OK-triangle*. If a triangle chain has at least one OK-triangle, then the chain is marked as an *OK-chain*.

The filtering sub-stage consists of two main stages, a stage of intermediate testing, and a post-processing stage.

Stage 1. At this stage determination of correctly triangulated regions is made. Each region is determined by movement of a front from a beginning point. As the beginning point an IC-point is chosen. Before processing the stage all the input points are marked as NT. For supporting movement of the front the next two lists of points are created: *BaseFront* and *ActiveFront*. The stage consists of the following steps:

1. At this step we search a beginning point for subsequent determination of the corresponding correctly triangulated island. For it we apply the umbrella-test sequentially for all the points having NT-status. According a result of this test a tested point gets one of the next statuses: FR, IC, BC, MC. If the point is determined as IC-point, then this point is accepted as a beginning point, and the testing of remaining NT-points is stopped. If a beginning point is not found, then this stage is aborted, and a jump to the stage of intermediate testing is made.
2. *BaseFront* and *ActiveFront* are initialized as empty lists. In *BaseFront* the found beginning point is added.
3. At this step we analyse all the points having connections with points of *BaseFront*. For each such point the following operations are made:
   3.1 If the point has IC-status, then consideration of this point is skipped, because it means, that the point was already passed by the front.
   3.2 If the point has status NT, then for this point the umbrella test is applied, and on the grounds of a result of the test the point gets one of the next statuses: FR, IC, BC, MC.
   3.3 If the point has IC-status, then the point is added in *ActiveFront*.
   3.4 If the point has MC-status, then the point's status is changed to CF.
4. If *ActiveFront* is empty, it means, that determination of the current island is done. In this case the next doings are made:
   4.1 If the determined island hasn't the external boundary (we obtain a local closed surface), then all the edges and triangles of the island are deleted, and all the island's points get status FR.
   4.2 A jump to step (1) to choice a new beginning point.
5. If *ActiveFront* is not empty, then we copy *ActiveFront* to *BaseFront*, and then we make *ActiveFront* empty.
6. A jump to step (3) to make next iteration of movement of the front.

The stage of intermediate testing. At this stage a testing results of stage 1 is made. If no one IC-point is found, then the filtering is aborted with an error code. If no one CF-point is found, then the filtering is ended with a code of success.

Stage 2. At this stage we try to normalize the points, which were detected at stage 1 as MC-points. For such points we detect and remove redundant triangles. Normalization is made by movement of a front too, and lists *BaseFront* and *ActiveFront* are used at this stage also. At description of the given stage the term "*releasing of a point*" is used. This term means, that all the triangles of a given point are removed and the point is marked as a FR-point. If as a result of elimination of a triangle we obtain a point without triangles, then this point is marked as a FR-point too. Movement of the front is realized as an iterative process, each iteration consist of the following steps:

1. A set of the points to test at the given iteration is determined. If it is the first iteration, then this set is made of all the points, which were marked

as CF-points at stage 1. For all subsequent iterations this set is made of all the points, which have status MC and have connections with points of *BaseFront*.

2. For each point of the determined set the next operations are made:
   2.1 All the OK-chains of triangles are determined.
   2.2 If there is only one OK-chain, then all other chains are removed.
   2.3 If there are several OK-chains, then the point is released, and its processing is ended.
   2.4 The point is marked as an IC-point (if the OK-chain is closed) or as a BC-point (if the OK-chain is open).
   2.5 If the point is determined as IC, then this point is added in *ActiveFront*.
3. If *ActiveFront* is empty, then the stage is ended.
4. *ActiveFront* is copied to *BaseFront*, then *ActiveFront* is made empty.
5. A jump to step (1) is made.

The stage of post-processing consists of the next steps:

1. All the remaining MC-points are realized. Thus, we obtain the set of points of correctly triangulated regions and the set of free points.
2. Boundaries of the correctly triangulated regions are determined.
3. For each boundary we determine if the boundary is the boundary of a hole or the boundary of an island. This determination is made on the basis of definition 2.1-2.

## 4. BASIC ISSUES OF STAGE B

### 4.1 Determination and construction of a TS

#### 4.11 Formalization of a TS

The basic operation of stage B is creation of a *triangle strip* (*TS*) in the non-reconstructed area. A TS connects two points (let's call them the *supporting points* of a TS) of a boundary (boundaries) of an island (islands). Reduction of a surface of class CS3 to a surface of class CS2 is made by creation of TSs, which make connections between islands (let's call such TSs *bridges*). Reduction a surface of class CS2 to a surface of class CS1 is also made by TSs, which are used for decomposition of complex holes.

As the basis a TS has a curve segment (let's call it *forming curve segment, FCS*) connecting the supporting points of the TS. The FCS is the segment (between the supporting points) of the curve formed by intersection of the original object surface and a plane (let's call this plane *secant plane, SP*) that passes through the supporting points. The surface of a TS is a strip of triangles approximating the original surface and having the FCS in the center.

## 4.12 Construction of a TS

With due regard for mentioned above, at stage B surface reconstruction is reduced to determination FCSs for the corresponding TSs. In non-reconstructed regions of a given A-surface the density of free points is essentially lesser then in SSRs. In addition, this density usually is very irregular. With due regard for these properties let's consider a method for determination of a FCS, that is based on analisys of the tension of a field. Let's suppose, that free points in a given neighborhood of a given SP are souses of a field (let's denote this field *G*), that has the following properties:

- The tension of the field is a scalar value. The field created by each point is spherically symmetrical. The tension of this field on a distance R from the point can be obtained by the formula:

  $$G = \frac{k}{R^2} \quad (4.12\text{-}1),$$

  where k – is a constant factor.
- In a given space point the result tension of the field is scalar sum of the tensions, which are created by each source of the field in the given point.
- This field can be shielded. If any shield exists, then the field of a given source effects in a given space point only if the source is visible from the given point.

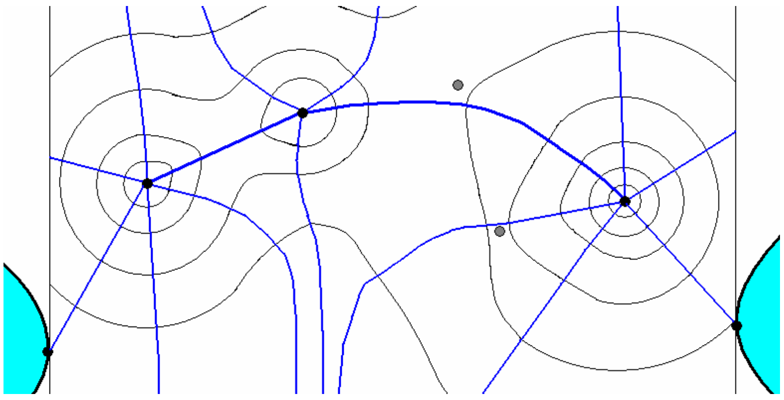


Figure 4.12-1

Let's consider the distribution of the tension of the field on a given SP (figure 4.12-1) created by a given set of free points. The projection of a given free point can be a local maximum of the tension (let's call such point on the plane *pole*, such projections are marked by black points) if this point is enough close to the SP. A remote free point (projections of these points are marked by gray points) usually has only influence on behavior of equipotential lines (marked by black lines).

Let's consider lines outgoing a given pole, such, the tangent line to which in each point is the perpendicular to the appropriate equipotential line. By analogy to the description of physical fields, let’s name these lines *force lines* of the field (marked by gray lines).

Force lines outgoing a given pole, it is possible to divide into the next two categories:

1) finite force lines binding poles with each other (marked by thick lines);
2) infinite force lines (marked by thin lines).

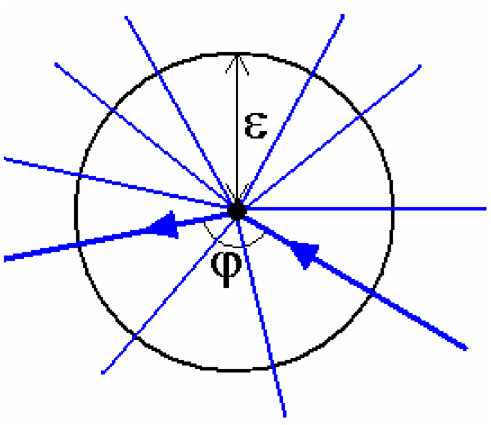


Figure 4.12-2

Let’s consider a circle with a small radius ε circumscribed around a given pole (figure 4.12-2). Let’s consider distribution of the tension of the field on this circle (for convenience, at this consideration we don’t consider the tension created by the pole). It is obvious, that there is a local maximum of the tension in the point of intersection of the circle and a force line of type (1).

As the FCS corresponding to a given pair of supporting points and a SP let’s consider a line, that is formed by force lines of type (1) linking poles and the force lines of type (2) linking the supporting points with the nearest poles. A FCS can be determined by sequentially tracking of these force lines.

Let's name a force line on which we came in a given pole, an *incoming force line*. Similarly, let’s name a line on which we leave a pole, an *outgoing force line*.

If topology of free points is complicated, a given pole can have more than one possible outgoing line. In this case we take as the outgoing force line the line having the maximum tension in the point of intersection with a given ε-circle, with condition that the angle (in figure 4.12-2 such angle is denoted as φ) between the tangent lines to the incoming and outgoing force lines in the pole is no smaller than a given threshold value.

**Lemma 4.12-1.** It is obvious, that the force lines form acceptable approximation of the FCS if for any two free points (*X* and *Y*) in a neighborhood of the FCS the following condition is satisfied: if $\{XX'\} < \{YY'\}$ then $x < y$, where $X'$, $Y'$ are the points on the FCS, which are the closest (along the surface) to *X* and *Y* correspondingly; *x* and *y* are the distances from correspondingly *X* and *Y* to the SP. Can be proved, that we have the most probability of observance of this condition when a given choice of the SP minimizes integral (2.2-1), where the supporting points as points *A* and *B* are taken, and the SP as the plane is taken.

In practice, at construction of a TS we use two auxiliary planes, which are parallel to a given SP. These planes are on the both sides from the SP, on some small identical distance (*d*) from it. For each of these planes we determine a broken line approximating the curve of intersection of the plane and the original surface by the described above algorithm for determination of a FCS. Then we connect vertices of the both broken lines to obtain triangles.

## 4.13 Conditions for successfully construction of a TS

Determination of the FCS of a TS is made on the basis of the method described in (4.12). Hence, the SP and the supporting points of the TS should be chosen taking into account the following conditions:

**Condition 4.13-1.** In accordance with lemma 4.12-1, integral (2.2-1) should be minimized.

**Condition 4.13-2.** The determined FCS should not pass through correctly reconstructed regions.

A SP can be defined by the two vectors:

- a *direction vector* ($\vec{D}$); this vector connects the supporting points of a given TS (let's denote them $P_1$ and $P_2$);
- an *orientation vector* ($\vec{O}$), it is some vector, that is not parallel to $\vec{D}$.

Let's consider, how we can satisfy condition 4.13-1. At determination of the SP only the surface normals in the supporting points (let's denote them $n_1$ and $n_2$ correspondingly) are known. Naturally, we need to minimize at least the sum of the angles between these normals and the SP. Let's replace the condition of the sum minimization by an equivalent condition of minimization of one angle. Let's define this angle (let's denote it $\varphi$) as the angle between the plane defined by vectors $\vec{D}$, $n_1$ and vector $n_2$. From the meant property of surface smoothness follows, that with reduction of the distance between $P_1$ and $P_2$ probability of observance of condition (4.13-1) increases.

Thus, a SP can be defined only by supporting points. As $\vec{O}$ the average vector of the external surface normals in these points is taken.

Also, let's note, that if $\vec{D}$ connects the boundaries of the two closest (in a given neighborhood) islands then condition 4.13-2 is satisfied.

## 4.2 Topological issues of connection of islands

**Definition 4.2-1.** Let we have an elastic weightless membrane that is a topological analog of a closed sphere circumscribed around an input point cloud. Let we have an A-surface of class CS3 (a set of islands) obtained from this point cloud. Let's consider the surface of the minimum of potential energy of this membrane with condition that the membrane fits closely to the surface of all the islands. Let's name such surface *islands-based surface* (*IS*).

**Definition 4.2-2.** Since each island is a part of a reconstructed surface of an object, then we can determine the external and the internal sides of an island. Let's name an island with determined sides *oriented* island. A method of determination of the orientation of an island will be described in (4.5). Let's define for an island the *positive direction* of movement along its external boundary the counter-clockwise direction, if we look to the external side. For the boundary of a hole inside an island the positive direction is reverse.

As it is described in (4.11), reduction of a surface of class CS3 to a surface of class CS2 is made by building connections (bridges) between islands with subsequent extraction of holes, which are made by the external boundaries of the islands and the built bridges. For consideration of topological issues of connection between islands, in the given paragraph let's consider a bridge only as the segment connecting the supporting points of the bridge.

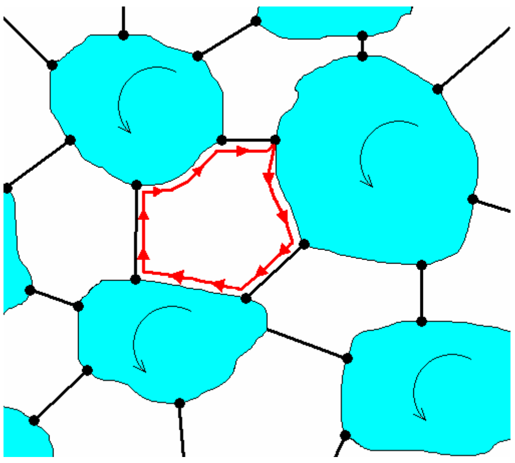


Figure 4.2-1

From a given topology of bridges (figure 4.2-1) we extract holes by the following algorithm:

1. A supporting point of a bridge as the beginning point ($P_0$) is chosen. The same point is accepted as the current point ($P_1$).
2. From $P_1$ we move along the corresponding bridge to its other supporting point.
3. From this supporting point we move along the boundary of the corresponding island in the positive direction until the first supporting point ($P_2$) of any bridge will be met.
4. If $P_2$ coincides with $P_0$, then the passed closed contour is considered as the found boundary of a hole, otherwise we accept $P_2$ as $P_1$ and return to step (2).

**Definition 4.2-3.** Let's name the number of bridges in the contour of a given hole the *rank* of the hole.

It is obvious, that a hole with a lesser rank has greater probability to be simple, than a hole with a greater rank.

Thus, a topology of connections between islands should satisfy to the following conditions:

**Condition 4.2-1.** The bridges should approximate the IS of the islands as precisely, as it is possible.

**Condition 4.2-2.** For avoiding a topological ambiguity, the supporting points of any two bridges should not coincide with each other.

**Condition 4.2-3.** The rank of the resulting holes should be minimized.

**Condition 4.2-4.** The number of the bridges should be minimized.

Condition 4.2-4 takes into account that a bridge is made by the enough costly method described in (4.12). Also, probability of generation of a wrong bridge increases with increasing the total number of generated bridges. But, this condition contradicts with condition 4.2-3. As a compromise between these conditions, let's minimize the total number of generated bridges to the limit that is necessary for existence of holes with rank 3 or greater. Can be easily proved the next lemma:

**Lemma 4.2-1.** In a given topology of bridges there are only holes with rank 3 or greater, if each island has no more than one connection with any other island. Besides, if each island is connected with all its nearest neighbors (along the original surface), then we have only holes with rank 3.

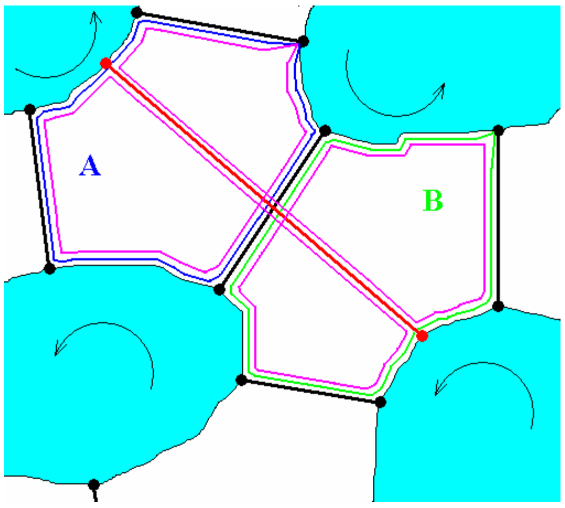


Figure 4.2-2

The described above method of reduction of a surface of class CS3 to a surface of class CS2 is sensitive to appearing wrong redundant bridges. As it is shown in figure 4.2-2, appearance of one wrong bridge leads to appearance of a contour having a complex topology. Such contour can't be interpreted as the boundary of a hole in a reconstructed surface.

**Lemma 4.2-2.** Let we have a topology created only by correct bridges. Thus, we have the set of correct contours of this topology. Let we add to the topology several wrong bridges with condition, that in each correct contour no more than one wrong bridge appears. Then, a closed contour containing a wrong bridge passes this bridge two times. Besides, if two correct contours connected by a wrong bridge have shared correct bridges, than these bridges are passed two times also.

**Proof.** Let's consider a wrong bridge, which connects two correct contours *A* and *B* (figure 4.2-2). The given wrong bridge takes away the trajectory of a round of contour *A* to contour *B*. It is obvious, that the trajectory can return to contour *A* only by the same wrong bridge. Also, for returning from contour *B* the trajectory should return to the supporting point of the wrong bridge on contour *B*. The trajectory can make it only if it makes round around all contour *B*. Similarly arguing, we make a conclusion, that for junction of the trajectory in the beginning point (according the condition of the hole extraction algorithm described above) the trajectory should make round around all contour *A*. Thus, if contours *A* and *B* have shared bridges, then these bridges are passed two times also.

Thus, if two correct contours haven't shared bridges, then we can simple detect a wrong bridge as a bridge passed two times. If these contours have shared bridges, the situation is more difficult, because we can't distinguish wrong and shared bridges. Taking into account that appearance of a wrong bridge is inadmissible, we consider that all the bridges passed two times should be removed.

An algorithm for holes extraction is described below. This algorithm has stability concerning appearance of wrong bridges, if such bridges don't break the condition of lemma 4.2-2. During holes extraction each segment of the boundary of an island made by supporting points of bridges can have two states – "free" and "used". Initially all the segments are marked as "free". If the contour of a hole is extracted successfully, then the segments of the contour are marked as "used". The algorithm consists of the following steps:

1. A beginning point for extraction of the contour of a new hole is chosen. Initially we choose a "free" segment, then the most "positive" endpoint of the segment is chosen (from the point of view of the positive direction for this segment) as the beginning point. If free segments are absent, the algorithm is ended.
2. Extraction of the contour of a hole from the determined beginning point is made. For extraction the described above algorithm is used. If during extraction any bridges passed two times are found then these bridges are deleted and a jump to step (1) is made.
3. The segments belonging to the found contour are marked as "used" and a jump to step (1) is made.

## 4.3 Determination of the placement of TSs at sub-stage CS3->CS2

### 4.31 Formulation of a sufficient condition

We need to choose pairs of the supporting points of TSs taking into account the conditions for a topology of connections between islands (4.2-1, 4.2-2, 4.2-3, 4.2-4) and the conditions for successful construction of a TS (4.13-1, 4.13-2).

Let's formulate a lemma that defines a sufficient condition for observance of conditions (4.13-1, 4.13-2), with assumption that the external boundaries of islands are smooth.

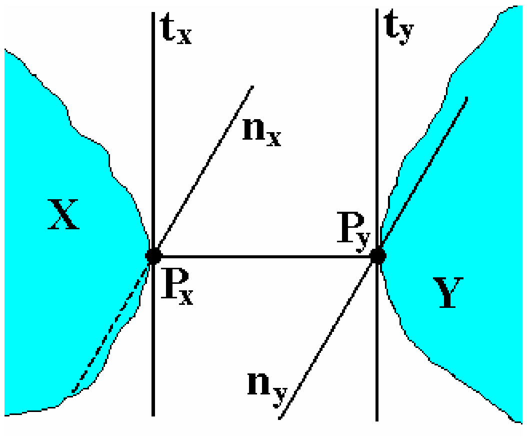


Figure 4.31-1

**Lemma 4.31-1.** Let we have the boundaries (*X* and *Y*) of two islands, and there are two supporting points ($P_x$ and $P_y$ correspondingly) on these boundaries. Let these points are the closest to each other points of boundaries *X* and *Y*. Let's denote the surface normals in $P_x$ and $P_y$ as $n_x$ and $n_y$, and the tangent lines to the boundaries in the given points as $t_x$ and $t_y$ correspondingly. So, if $t_x$ and $t_y$ are parallel, then vectors $\overrightarrow{P_xP_y}$ , $n_x$, $n_y$ lie in the same plane.

**Proof.** In a given point of a boundary the tangent line lie in the plane approximating the reconstructed surface in the given point. Thus this tangent line is perpendicular to the surface normal in the given point. The tangent line is an approximation of the boundary in some close neighborhood of the given point. As is known, the shortest distance between two parallel straight lines is a perpendicular to them. Thus, we have the three vectors $\overrightarrow{P_xP_y}$ , $n_x$, $n_y$, which are perpendicular to the same straight line (one of the tangent lines). Therefore, these vectors lie in the same plane.

## 4.32 Definition of field H

Let along the boundary (boundaries) of each island, in the positive direction, a current of some nature flows. For each boundary the current has the same intensity. Let interaction between two elementary currents $\vec{I}_1$ and $\vec{I}_2$ placed in points $X_1$ and $X_2$ correspondingly is defined by the formula:

$$\vec{F}_{12} = -\delta k \frac{\vec{d}_{12}(\vec{I}_1 \vec{I}_2) Sin\alpha}{A + r^2} \quad (4.32\text{-}1),$$

where

$\vec{F}_{12}$ is the force having effect on the current $\vec{I}_1$;

$\delta$= 0, if $\vec{I_1}$ and $\vec{I_2}$ belong to a boundary (boundaries) of the same island, and $\delta$= 1 otherwise; let's add also the next condition of shielding: $\delta$= 1, if $\vec{I_1}$ and $\vec{I_2}$ are visible for each other, and $\delta$= 0 otherwise;
*k* is a constant factor;
$\vec{d_{12}}$ is the unit vector directed from *X₁* to *X₂*;
$\alpha$ is the angle between vectors -$\vec{d_{12}}$ and $\vec{I_2}$;
*r* is the distance between *X₁* and *X₂*;
*A* is a small constant used for prevention of appearance of very large interaction forces when $r \to 0$;
the minus in the beginning means, that opposite directed currents attracts.

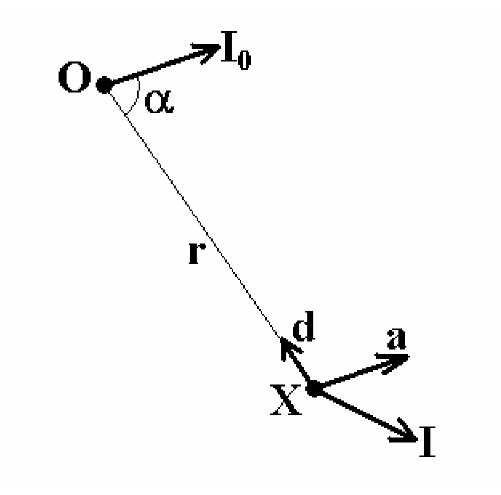


Figure 4.32-1

Such interaction can be provided by a field (let's denote it *H*) created by each elementary current. Let we have in a given space point (*O*) an elementary current ($\vec{I_0}$). Let the tension of the field created by $\vec{I_0}$ in other space point (*X*) is defined by the two following vectors (figure 4.32-1):

$\vec{d}$ is the unit vector directed from *X* to *O*; this vector defines the direction of the force having effect on a current in *X*;
$\vec{a}$ is the vector, that is responsible for the volume and the sign of the interaction force; this vector has the same direction that $\vec{I_0}$ and is defined by the next formula:

$$\vec{a} = -k\frac{\vec{I_0}Sin\alpha}{A+r^2} \quad (4.32\text{-}2),$$

where $\alpha$ is the angle between vectors $\vec{I_0}$ and -$\vec{d}$.

In this case the force having effect on a current ($\vec{I}$) placed in *X* is defined by the next equation:

$$\vec{F} = \delta\vec{d}(\vec{I}\vec{a}) \text{ (4.32-3)}$$

The field defined by this way has the next important property: the straight line of the vector of a force having effect on an elementary current does not depend on the direction of the current and is defined only by vector $\vec{d}$ of the tension of the field in the point of location of the current. As a force line of the field we consider a curve, in each point of which the tangent line is the line of a force vector in the same point.

Let's assume, that the interaction defined by formula 4.32-1 has the superposition property. For realization of this property we need to define an operation of summation for the tension of the field H, with the condition that for any current $\vec{I}$ the next equations would be true:

$$\vec{F}_1 = \vec{I}H_1$$
$$\vec{F}_2 = \vec{I}H_2$$
$$\vec{F} = \vec{F}_1 + \vec{F}_2 = \vec{I}(H_1 + H_2)$$
$$H_1 + H_2 = H_2 + H_1$$
$$(H_1 + H_2) + H_3 = H_1 + (H_2 + H_3)$$

Because the tension of the field H created by an elementary current in a given space point is defined by the two independent vectors, the field H with the defined operation of summation of the tensions is a tensor field. The tension of this field created by an elementary current in a given space point can be represented by the next matrix:

$$H = \begin{pmatrix} d_x a_x & d_x a_y & d_x a_z \\ d_y a_x & d_y a_y & d_y a_z \\ d_z a_x & d_z a_y & d_z a_z \end{pmatrix} \text{(4.32-4),}$$

where $d_{x,y,z}$, $a_{x,y,z}$ are components of the corresponding vectors $\vec{d}$ and $\vec{a}$.

In this case formula 4.32-3 can be written as:

$$\vec{F} = \delta\vec{I}H \text{ (4.32-5)}$$

## 4.33 A method of determination of the supporting points of a TS

From the properties of the interaction defined by formula 4.32-1 follows, that the value of the attraction force along the boundary of an island increases with completeness of accomplishment of the condition of lemma 4.31-1 and reduction of the distance up to the external boundaries of neighbor islands. Let suppose that there is a local maximum of the force in a point $P_1$ of a given segment of the boundary of an island. Let we track the force line of the field that is outgoing from $P_1$ in the direction of the force vector. Let's denote the found second endpoint of this force line as $P_2$. $P_1$ and $P_2$ can be considered as the supporting points of a possible bridge between the corresponding islands. The superposition property of the field H provides, that the behavior of force lines of the field satisfies to condition 4.13-2. Because of the same property the boundaries of interacted islands are presented to each other as smooth lines according to the condition of lemma 4.31-1.

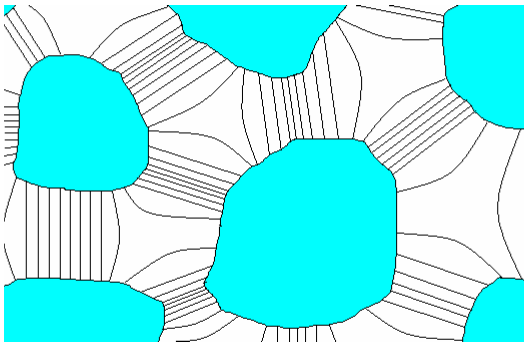


Figure 4.33-1

Thus, let we have the set of all the point pairs (let's denote this set *{S}*) formed by the described above method. Then, for stage CS3->CS2 the set of pairs of the supporting points of a TS is the subset of *{S}* constructed with observance of conditions 4.2-1, 4.2-2, 4.2-3, 4.2-4.

## 4.4 Determination of placement of a TS at sub-stage CS2->CS1

Decomposition of a complex hole is made by recursive subdivision of the hole. The subdivision is made by creation of a TS inside the hole contour. The supporting points of this TS is chosen generally by the same way that for stage CS3->CS2. But for this stage there are the following differences:

- Each hole is considered separately, and only the current flowing along the hole's boundary is taken into consideration. Hence, the direction of the current can be chosen at will.
- In formula 4.32-1 term $\delta$ is equal to 0 if the points of placement of elementary currents $\vec{I}_1$ and $\vec{I}_2$ is connected by the shortest segment of the boundary, that is shorter than $L/N$; where $L$ is the length of the boundary, $N \geq 2$. This condition is applied to avoid appearance of local maximums of the attraction force in sharp corners of the boundary. The shielding condition for $\delta$ is the same.

Taking into account the described above modifications, the created TS splits the hole in the narrowest place, where corresponding segments of the boundary lie approximately in the same plane, with condition, that the smallest child hole has the length of the contour not lesser than $L/N+d$, where *d* is the length of the FCS of the TS.

## 4.5 Determination of the orientation of islands

As was mentioned in (4.2, 4.3), one of the key things of stage CS3->CS2 is determination of the orientation of an island.

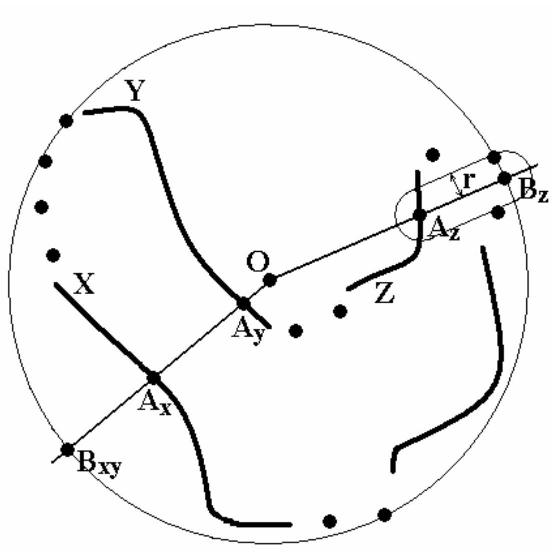


Figure 4.5-1

One of possible methods is described below.

Around an input set of points let's circumscribe the minimal sphere, let's denote the center of the sphere *O* (figure 4.5-1). Let *A* is a point of the surface of an island and *B* is the point of intersection of the sphere and half-line *OA*. The direction of the normal in *A* making a positive dot product with vector $\overrightarrow{OA}$ we consider as external if the following conditions are observed:

1) Segment *AB* don't intersect the surface of any other island;
2) There are no free points on a distance lesser than *r* from segment *AB*; $r = kd$, where *d* is the typical distance between free points in a given region, *k* is a constant factor.

If such point *A* is found for an island, then its orientation can be determined by this way. However, this method (or other simple method) is suitable not for every case. For example, in figure 4.5-1 the method is working for island *X* and is not working for islands *Y* and *Z*.

Let we have such "difficult" island in an encirclement of correctly oriented islands (let's call them *standard islands*). For determination of the orientation of this island let's use the fact, that attraction predominates between correctly oriented neighbor islands at using field H.

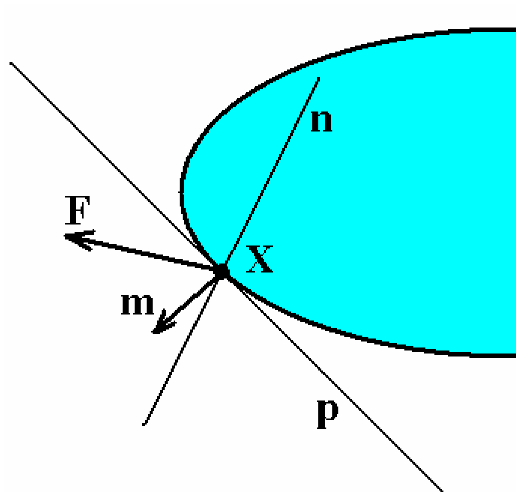


Figure 4.5-2

For the island, for the both possible variants of its orientation, let's calculate the integral:

$$\oint_L \vec{F}\vec{m}dl \quad (4.5\text{-}1),$$

where

*L* is the contour of the external boundary of the island;

$\vec{F}$ is the force vector in a point (*X*) of the boundary of the island;

$\vec{m}$ is the unit vector (figure 4.5-2) in the same boundary point, which is: 1) perpendicular to the normal (*n*) of the island surface in the point; 2) perpendicular to the tangent line (*p*) of the boundary in the point; 3) directed to the non-reconstructed area.

For the correct orientation of the island we have the greatest value of integral 4.5-1.

## 4.6 Additional issues of modeling of the fields

Using each of the defined above fields (G and H) we can obtain more adequate space distribution of the field tension, if we use shielding. In accordance with the defined properties of the fields, the shielding can be modeled by using existing technologies for visualization. It is obvious, that regions of the reconstructed surface should shield the both fields.

Also, several effective optimizing conditions for the modeling are described below:

**Condition 4.6-1.** Let's consider a point (*X*) on the boundary of an island (see figure 4.5-2 and the description of formula 4.5-1). The plane defined by *n* and *p* divides the space into two subspaces. Let's consider that field H created by an elementary current $\vec{I}$ placed in *X* exists only in the subspace that contains the endpoint of $\vec{m}$.

**Condition 4.6-2.** At construction of a TS we consider only free points inside the parallelepiped formed by:

- two planes, which are parallel to the SP of the TS and are on the both sides from the SP, on the same distance (*l*) from it;
- two planes, which are perpendicular to the SP and parallel to $\vec{D}$; one plane is placed on an established distance (*h*) "above" $\vec{D}$, and other plane is placed on an established distance (*d*) "below" $\vec{D}$; the "top" direction is defined by a vector, that is perpendicular to $\vec{D}$ and has positive dot products with the vectors of external normals in the supporting points of the TS.

The values *l*, *h*, *d* usually depend on $|\vec{D}|$.

## 4.7 A method of detection of a simple hole

At the beginning let's define a projection plane of a given hole. As this plane we take the own plane of the hole. Then for each vertex of the boundary (the boundary is considered as a broken-line) we define a corresponding auxiliary point (let's call it *external contour point, ECP)* in the following way:

1) we calculate vector $\vec{m}$ introduced in formula 4.5-1;
2) we calculate vector $\vec{p}$ by the formula: $\vec{p} = -\varepsilon\vec{m}$, where ε is a small positive value;
3) the endpoint of $\vec{p}$ is considered as the ECP of the given vertex.

Thus, if the line of hole's boundary projection don't cross itself, and all the projections of ECPs lie outside the region, restricted by this line, we consider, that the hole has unambiguous projection. In figure 4.7-1 a case of ambiguous projection of a hole is shown, when several projections of ECPs (featured by circles) are inside the area of the projection of the hole.

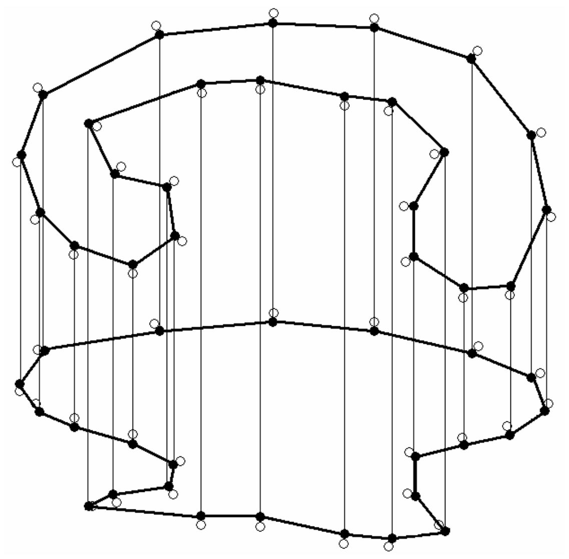


Figure 4.7-1

## 5. ALGORITHMS OF IMPLEMENTATION OF STAGE B

### 5.1 Sub-stage CS3->CS2

The described below algorithm has five stages.

Stage 1. At this stage we determine the orientation of all the islands by the method described in (4.6).

Stage 2. At this stage a list of pairs of the supporting points of possible bridges between the islands is created (let's denote them *BridgePointsList*). But generally, this list is not completely done at this stage.

1. The list of all the points of the boundaries of all the islands having a local maximum of the attraction force (let's denote this list *MaxAttractionPointsList*) is made.
2. *MaxAttractionPointsList* is sorted in decreasing order values of the attraction force in the points.
3. If *MaxAttractionPointsList* has less than two elements, then this stage is ended.
4. The first point of *MaxAttractionPointsList* (let's denote this point $P_1$) is chosen.
5. The force line outgoing from $P_1$ in the direction of the force vector is tracked and the other endpoint of this force line (let's denote it $P_2$) is determined.
6. If $P_2$ is not an element of *MaxAttractionPointsList*, then $P_1$ is removed from the list and a jump to step (3) is made.

7. If there is no pair of points representing the same islands that pair ($P_1$,$P_2$) in *BridgePointsList*, then pair ($P_1$,$P_2$) is added to the list.
8. Points $P_1$ and $P_2$ are removed from *MaxAttractionPointsList* and a jump to step (3) is made.

Stage 3. At this stage we determine missing elements of *BridgePointsList* for islands having less than 3 possible bridges.

1. The list of the islands having less than 3 possible bridges (let's denote it *UnsuffLinkedIslandsList*) is made;
2. If *UnsuffLinkedIslandsList* is not empty, then the first element of the list is chosen (let's denote this island *I*), else the stage is ended.
3. The list of all the boundary points of *I* with exception of points assisting in *BridgePointsList* (let's denote this list *IslandBoundaryPointsList*) is made, then this list is sorted in decreasing order values of the attraction force.
4. The list of the boundary points of *I* in decreasing order values of the attraction force in the points (let's denote it *CandidateBridgePointsList*) is made. We sequentially consider elements of *IslandBoundaryPointsList* and add in *CandidateBridgePointsList* only points, which satisfy to the next condition: the length of the shortest boundary segment between a given considered point and any point from *CandidateBridgePointsList* or *BridgePointsList* is greater than some threshold value.
5. If *CandidateBridgePointsList* is empty, then *I* is removed from *UnsuffLinkedIslandsList* and a jump to step (2) is made.
6. The first element in *CandidateBridgePointsList* (let's denote this point $P_1$) is chosen.
7. The force line outgoing from $P_1$ in the direction of the force vector is tracked and the other endpoint of this force line (let's denote it $P_2$) is determined.
8. If $P_2$ assists in *BridgePointsList*, then a jump to step (10) is made.
9. Pair of points ($P_1$,$P_2$) is added in *BridgePointsList*.
10. If pair ($P_1$,$P_2$) describes a 3-rd possible bridge for any island(s) from *UnsuffLinkedIslandsList*, then this island(s) is removed from the list. If *I* is one of such islands, then a jump to step (2) is made.
11. $P_1$ is removed from *CandidateBridgePointsList*, and a jump to step (4) is made.

Stage 4. At this stage, on the base of created *BridgePointsList* extraction of holes and detection of wrong bridges is made by the methods described in (4.2). If an element of *BridgePointsList* describes a wrong bridge, then this element is removed from the list.

Stage 5. For each pair of the supporting points of *BridgePointsList* the corresponding TS is made by the method described in (4.12).

## 5.2 Sub-stage CS2->CS1

**Definition 5.2-1.** Let's name a given simple hole a *simple flat hole*, if the distance between any point of its boundary and its own plane does not exceed a threshold limit *h*; $h = kD$, where *D* is the length of the longest chord of the boundary of the hole, *k* is a constant factor.

**Condition 5.2-1.** For more effective work of subsequent stage C, let's demand that all holes on the output of this sub-stage should be simple flat holes.

Generally, this sub-stage is carried out by separate processing each hole obtained at sub-stage CS3->CS2. Each input hole is processed by recursive application of the described below algorithm:

1. If the given hole is a simple flat hole, then its processing is ended.
2. We consider a current flowing along the boundary of the hole and determine distribution of the interaction force on the boundary.
3. The point having the maximum of the force (let's denote this point $P_1$) is found on the boundary.
4. The force line outgoing from $P_1$ in the direction of the force vector is tracked and the other endpoint of this force line (let's denote this point $P_2$) is determined.
5. A TS is made with using $P_1$ and $P_2$ as its supporting points.

## 6. STAGE C

At this stage surface reconstruction inside holes obtained at stage B is made. Because we have deal only with simple flat holes, surface reconstruction can be made by a simple and fast method. For each hole we reduce the task of surface reconstruction inside it to the task of 2D triangulation inside the area that is made by projection of the boundary of the hole onto the own plane of the hole. For this triangulation a variant of the classical GT-algorithm [BE95] is applied, but as the length of the edge between a given pair of 2D-points the distance between the corresponding 3D-points is used.

Stage C consists of two sub-stages.

Sub-stage 1. At this sub-stage the input parameters for the 2D-triangulation algorithm are determined for each hole. Initially, for each hole the own plane and the line of projection the boundary onto this plane (let's call this line *2D-contour*) are determined. Then, for each hole the corresponding set of free points considered for surface reconstruction inside the hole is determined (let's denote this set *{P}*). A free point belongs to *{P}* of a given hole, if its projection onto the own plane of the hole is inside the 2D-contour. If a free point can be referred to such sets of several holes simultaneously, then the point is referred to the set of the hole that has closest own plane to the given point.

Sub-stage 2. At this sub-stage surface reconstruction is carried out separately for each hole. For a hole the algorithm of 2D-triangulation is applied with using the input data obtained at the previous sub-stage.

## 7. SOME RESULTS

At present there is an implementation of strategy S2, in this implementation stages A and C are implemented by prototypes of the described above corresponding algorithms. Stage B is implemented by the simplified algorithms described in [ES02]. Several experiments with using the given implementation were made. The results for several samples are shown in Table 7-1. All the timing measurements were made on a PC with 1500 MHz Athlon and 1GB main memory.

“Bunny” (figure 7-1) and “Bone” (figure 7-2) are widely used for testing models. Also we use two artificially damaged variants of "Bunny" to test. In several regions points were removed and in several regions points density was decreased 10 times. Model “Bunny1” is a little damaged, and model “Bunny'2” – heavy. In figures 7-3a, 7-4a correspondingly the intermediate A-surfaces obtained for these models are shown. In figures 7-3b, 7-4b the same models are shown after completion of surface reconstruction. The “Face” is a fragment of a distance scanned big sculpture. In figure 7-5a the input point cloud and in figure 7-5b the reconstructed and lighted surface are shown.

“Bunny”, “Bone” belong to class CP1, and “Bunny1”, “Bunny2”, “Face” belong to class CP2. The obtained results can be grouped in the next free groups:

1. Results for point clouds of class CP1, which have a good sampling quality (“Bunny”). An obtained A-surface belongs to class CS1 and has a relative little number of holes. In comparison with [ACD00, DG01] we have sufficiently better speed.
2. Results for point clouds of class CP1, which have an average sampling quality (“Bone”) and clouds of class CP2, which have a little share of defects (“Bunny1”). An obtained A-surface also belongs to class CS1, but the total area of holes is essentially greater than for group (1). We have speed that is comparable with [DG01].
3. Results for point clouds of class CP2, which have serious defects. An obtained A-surface belongs to class CS2 (“Bunny2”) or to class CS3 (“Face”). We can reconstruct a closed surface from such clouds, when available algorithms show unsatisfactory results ("Bunny2") or can't be applied ("Face"). However, in especially difficult cases (in figure 7-4b marked by a circle) we still can’t make surface reconstruction correctly.

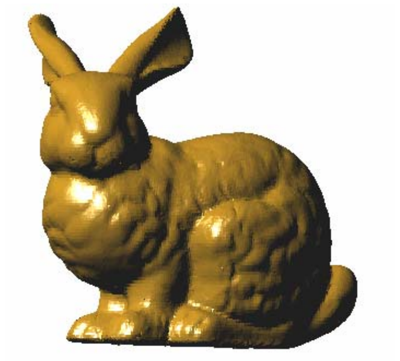


Figure 7-1

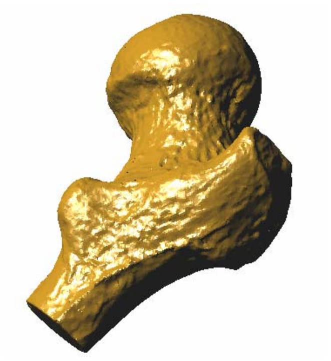


Figure 7-2

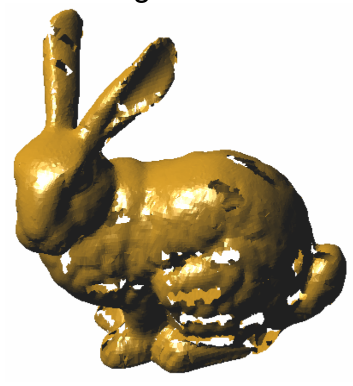


Figure 7-3a

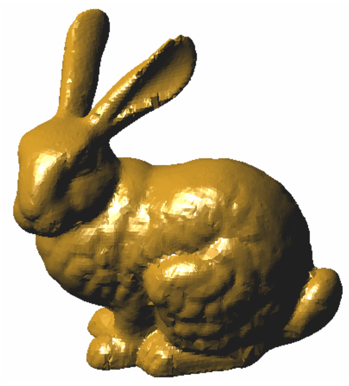


Figure 7-3b

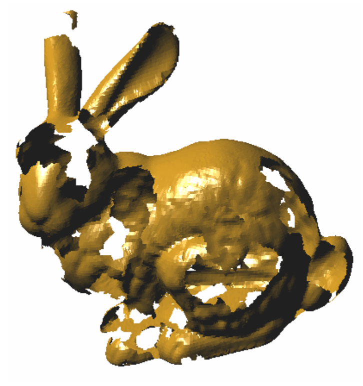


Figure 7-4a

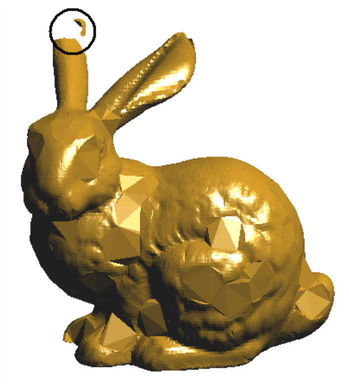


Figure 7-4b

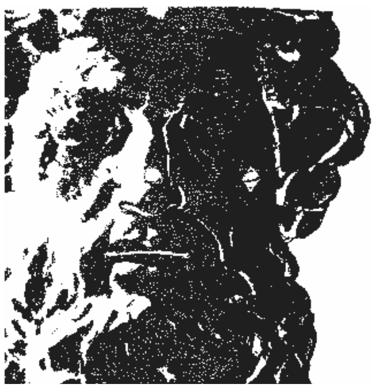


Figure 7-5a

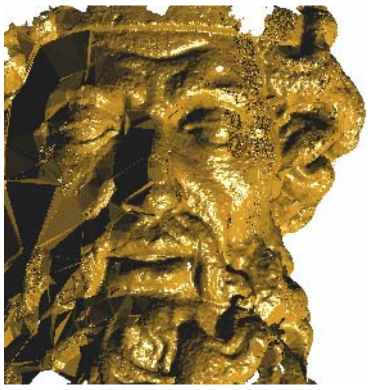


Figure 7-5b

| Sample | N points | $U_a$ | $U_m$ | Time (sec) | Speed (point/sec) |
|---|---|---|---|---|---|
| Bunny | 35947 | 1.7 | 2.9 | 3.6 | 9985 |
| Bone | 68537 | 3.2 | 7.1 | 15.1 | 4539 |
| Bunny1 | 29491 | 1.9 | 28.9 | 14.2 | 2077 |
| Bunny2 | 26553 | 2.2 | 72.4 | 20.9 | 1270 |
| Face | 232511 | 2.1 | 159.2 | 385.8 | 603 |

Table 7-1

## 8. REFERENCES

[ACD00] N. Amenta, S. Choi, T. K. Dey, N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In Proc. of 16th Sympos. Comput. Geom., 2000, pp.213-222.

[AGJ02] U. Adamy, J. Giesen, M. John. Surface Reconstruction using Umbrella Filters. Computational Geometry Theory & Applications 21(1-2): pp.63-86, January 2002.

[AS96] Maria-Elena Algorri and Francis Schmitt. Surface reconstruction from unstructured 3d data. Computer Graphics forum, 15(1):47-60, 1996.

[BB97] 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 Sciences, Purdue University, 1997.

[BBC97] 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.

[BBX95] 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.

[BBX97] Chandrajit Bajaj, Fausto Bernardini, and Guoliang Xu. Reconstructing surfaces and functions on surfaces from unorganized 3d data. Algorithmica, 19:243-261, 1997.

[BE95] M. Bern, D.Eppstein. Mesh Generation and Optimal Triangulation. Computing in Euclidean Geometry, D.-Z. Du and F.K. Hwang, eds., World Scientific, 1992, 2nd edition, 1995.

[BH93] 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.

[BH94] 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.

[BS96] 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.

[BTG95] 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.

[DA97] Dominique Attali. r-regular shape reconstruction from unorganized points. In ACM Symposium on Computational Geometry, pages 248-253, 1997. Nice, France.

[DG01] T. K. Dey, J. Giesen. Detecting undersampling in surface reconstruction. Proc. 17th Sympos. Comput. Geom., 2001, pp.257-263.

[DH73] Richard O. Duda and Peter E. Hart. Pattern Classification and Scene Analysis. Wiley and Sons, Inc., 1973.

[EPM92] Herbert Edelsbrunner and Ernst Mucke. 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.

[EPM93] Ernst Peter Mucke. Shapes and implementations in three-dimensional geometry. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, September 1993.

[ES02] A. Emelyanov, V. Skala. Surface reconstruction from problem point cloud. In Proc. of Graphicon’02, pp.31-37.

[FW97] 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.

[GMW97] Baining Guo, Jai Menon, and Brian Willette. Surface reconstruction using alpha-shapes. Computer Graphics forum, Vol. 16(No. 4):177-190, October 1997.

[HDD92] 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.

[HDD93] 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.

[HE92] Herbert Edelsbrunner. Weighted alpha shapes, 1992. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[HH94] Hugues Hoppe. Surface Reconstruction from Unorganized Points. PhD thesis, Univ. of Washington, Seattle WA, 1994.

[HL79] 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.

[HJL93] Josef Hoschek and Dieter Lasser. Fundamentals of Computer Aided Geometric Design. A.K. Peters, 1993.

[IBS97] 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.

[JB84] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. ACM Transactions on Graphics, 3(4):266-286, October 1984.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. Computer Graphics, 21(4):163-169, July 1987.

[MBL91] 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.

[MM97] Robert Mencl and Heinrich Muller. Graph-based surface reconstruction using structures in scattered point sets. Research Reports No. 661, 662, 1997, Fachbereich Informatik, Lehrstuhl VII, University of Dortmund, Germany.

[MV01] Michal Varnuska. Surface reconstruction of geometrical objects from scattered point data. University of West Bohemia, 2001.

[PS85] Franco P. Preparata and Michael Ian Shamos. Computational Geometry: An Introduction. Springer Verlag, 1985.

[RM95] Detlef Ruprecht and Heinrich Muller. Spatial free form deformation with scattered data interpolation methods. Computers and Graphics 19, pages 63-71, 1995.

[RV94] Remco C. Veltkamp. Closed object boundaries from scattered points. In Lecture Notes in Computer Science 885. Springer Verlag, 1994.

[RV95] Remco C. Veltkamp. Boundaries through scattered points of unknown density. Graphics Models and Image Processing, 57(6):441-452, November 1995.

[RGW97] Gerhard Roth and Eko Wibowoo. An efficient volumetric method for building closed triangular meshes from 3d image and point data. In Graphics Interface'97, pages 173-180, 1997.

[SB97] Thomas Schreiber and Guido Brunnett. Approximating 3d objects from measured points. In Proceedings of 30th ISATA, Florence, Italy, 1997.

[SM91] Shigeru Muraki. Volumetric shape description of range data using "blobby model". Computer Graphics, pages 217-226, July 1991. Proceedings of SIGGRAPH'91.

[ST92] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. Computer Graphics, 26:185-194, July 1992. Siggraph '92 Proceedings.

[TM91] 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.

[TS97] Thomas Schreiber. Approximation of 3d objects. In Proceedings of the 3rd Conference on Geometric Modeling, Dagstuhl, Germany, 1997.

[TWK88] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. Artificial Intelligence, 36:91-123, 1988.

## 9. PUBLICATIONS

A. Emelyanov, V. Skala. Surface reconstruction from problem point cloud. In Proc. of Graphicon'02, pp.31-37.

A. Emelyanov, V. Skala. An algorithm for problem point cloud processing. In Proc. of VEonPC'02, pp.68-76.