

# Reconstructing Surfaces and Functions on Surfaces from Unorganized Three-Dimensional Data <sup>\*</sup>

C. L. Bajaj   F. Bernardini <sup>†</sup>   G. Xu

Department of Computer Sciences  
Purdue University  
West Lafayette, Indiana

{bajaj,fxb,xuguo}@cs.purdue.edu, Tel: 317-494-6531

## Abstract

Creating a computer model from an existing part is a common problem in Reverse Engineering. The part might be scanned with a device like the laser range scanner, or points might be measured on its surface with a mechanical probe. Sometimes, not only the spatial location of points, but also some associated physical property can be measured. The problem of automatically reconstructing from this data a topologically consistent and geometrically accurate model of the object and of the sampled scalar field is the subject of this paper.

The algorithm proposed in this paper can deal with connected, orientable manifolds of unrestricted topological type, given a sufficiently dense and uniform sampling of the object's surface. It is capable of automatically reconstructing both the model and a scalar field over its surface. It uses Delaunay triangulations, Voronoi diagrams and alpha-shapes for efficiency of computation and theoretical soundness. It generates a representation of the surface and the field based on Bernstein-Bézier polynomial implicit patches (A-patches), that are guaranteed to be smooth and single-sheeted.

## 1 Introduction

Computer Aided Design (CAD) and Engineering (CAE) are being extensively used in all the phases of the manufacturing process. However, creating a CAD model of an existing part still requires an extremely time-consuming manual data-entry process.

Large scale reverse-engineering processes are nowadays commonplace in the manufacturing industry. For example, a company that manufactures mechanical parts might have on stock parts for which a CAD model is not available. In designing new parts, the company would like to use one of the existing parts as a base model, and try to improve it. As another example, imagine that an experiment was conducted on a prototype and data was measured at scattered points on its surface. For example, a jet engine was operated in variable conditions in a wind tunnel, and the temperature at points on its surface sampled over time.

Several types of sensors are available to scan a 3D object and measure the location of points on its surface. Mechanical probes, used in the manufacturing industry, are extremely accurate but slow to use. Laser range scanners can measure the location of a large number of points in a relatively short time. Recent models are also capable of

---

<sup>\*</sup>Supported in part by NSF grants CCR 92-22467, DMS 91-01424, AFOSR grants F49620-93-10138, F49620-94-1-0080, ONR grant N00014-94-1-0370 and NASA grant NAG-93-1-1473.

<sup>†</sup>Additional partial support from CNR, Italy.

capturing the RGB components of the color of the surface at each sample point. Less accurate (and cheaper) hand-held devices, based on ultrasound or magnetic fields, are also being marketed. When the geometry of the scanned object is simple, for example flat or cylindrical, the points are organized into a rectangular grid, and a complete, unambiguous model of the object is readily available. However, except for such particularly simple shapes, the scanning process must be repeated from different points of view, and the results merged together [47]. In other cases, for example scattered measurements with a digitizing stylus, there is no ordering in the data.

The problem of reconstructing a topologically consistent and geometrically accurate CAD model of the object and of the sampled physical properties from the scanned data points is the subject of this research. The approach one might take in solving this problem depends heavily on the assumptions that can be made about the sampled object and the data itself. For example, the object might be assumed to be smooth or instead containing creases and corners. The sampling might be very dense and uniform, or rather sparse.

Last-generation devices are capable of measuring  $10^4 \sim 10^5$  points per second, with a resolution of  $10^{-4}$  inches. The problem is therefore not that of inferring a reasonable shape from a set of sparse points on the object's surface, but rather that of providing a compact, usable, accurate and topologically consistent representation of the object from a dense sampling of its surface.

Our research has focused on reconstruction algorithms based on piecewise algebraic surfaces [6, 5, 15]. An algebraic surface [46] is defined as the two-dimensional algebraic variety expressed by the equation  $f(x, y, z) = 0$ , where  $f$  is a polynomial. A piecewise algebraic surface is a collection of surface *patches*, pieced together with some degree of derivative continuity. Each patch is an algebraic surface with a finite extent, usually given by a bounding *box* or *tetrahedron*. Piecewise algebraic surfaces of low degree have many attractive qualities, from the closure properties with respect to important modeling operations such as intersection and blending, to a high design flexibility for a relatively low algebraic degree [4]. In our reconstruction algorithm, we use barycentric Bernstein-Bézier patches that satisfy certain smoothness conditions (*A-patches*, see [7]). We consider the following reconstruction problem:

Let an unorganized collection of points  $P = \{(x_i, y_i, z_i)\} \subset \mathbf{R}^3$  and associated values  $V = \{v_i\} \subset \mathbf{R}^1$ ,  $i = 1, \dots, n$ , be given. The points  $P$  are assumed to be sampled from a domain  $D$  in  $\mathbf{R}^3$  (the boundary of a three-dimensional object, that is a two-manifold without boundary) while the values  $V$  are assumed sampled from some scalar function  $F$  on the domain  $D$ .

Construct a  $C^1$  smooth piecewise-polynomial surface  $S^D : f^D(x, y, z) = 0$  and a  $C^1$  smooth piecewise polynomial function (function-on-surface)  $S^F : f^F(x, y, z)$  on some domain that contains  $S^D$  such that, for  $i = 1, \dots, n$ :

1.  $|f^D(x_i, y_i, z_i)| < \varepsilon^D$ ,
2.  $|f^F(x_i, y_i, z_i) - v_i| < \varepsilon^F$ ,

where  $\varepsilon^D$  and  $\varepsilon^F$  are user-defined approximation parameters.

## 2 Related work

Reconstructing the domain surface  $S^D$  from unorganized points in  $\mathbf{R}^3$  is a fundamental problem in CAD and computer vision. Techniques for piecewise-linear reconstruction from unorganized points are described in [42, 17, 18, 48, 34, 47, 20]. The reconstructed triangle mesh can be used as a starting point for a successive parametric or subdivision surface fitting, as in [33, 24, 35].

In this paper we construct a  $C^1$  smooth domain surface  $S^D$  using piecewise cubic, implicit Bézier patches (the zero contour of a  $C^1$  trivariate piecewise Bézier function). These implicit Bézier patches are guaranteed to be single-sheeted within each tetrahedron. Related prior work includes [2, 7, 8, 21, 22, 31, 32, 39, 37]. The surface fitting paper of [36] is similar to ours in that it only assumes a sufficiently dense set of input data points but differs from our approach in the adaptive nature of refinement, in time efficiency, and in the degree of the implicit surface patches used. Paper [36] uses an octree-like subdivision scheme and triquadratic (degree six) tensor product implicit surface patches with a Powell-Sabin-type split to achieve  $C^1$  continuity. The Powell-Sabin split produces 64 pieces per cube. Our scheme effectively utilizes the incremental Delaunay triangulation for a more adaptive fit, the dual Voronoi diagram for efficient point location in signed distance computations and degree three implicit surface patches. Furthermore, at the same time it also computes a  $C^1$  smooth approximation  $S^F$  of the sampled scalar function.

If the surface  $S^D$  is given, the problem of constructing the scalar function  $S^F$  is known as function interpolation on a surface, and arises in several application areas, for, e.g., in modeling and visualizing the rainfall on the earth, the pressure on the wing of an airplane, or the temperature on a human body. Note that the trivariate scalar function  $S^F$  is a two dimensional surface in  $\mathbf{R}^4$ , since its domain is the two dimensional surface  $S^D$  (and not all of  $\mathbf{R}^3$ ). The problem is relatively recent and was posed as an open question by Barnhill [10]. A number of methods have been developed since then for dealing with the problem (for surveys see [11, 30, 41, 38]). Most of the solutions interpolate scattered data over planar or spherical domain surfaces. In [13] and [39], the domain surface is generalized to a convex surface and a topological genus zero surface, respectively. Pottmann [43] presents a  $C^1$  method which does not possess similar restrictions on the domain surface but requires it to be at least  $C^2$  differentiable. In [12] the  $C^2$  restriction is dropped, however, the interpolation surface is constructed by transfinite interpolation using nonpolynomials. A similar nonpolynomial transfinite interpolant construction is used in [40], while in [45] at least  $C^4$  is required for interpolation. Bajaj and Xu [9] present a  $C^2$  scheme using quintic polynomials to reconstruct the scalar field defined on a smooth curved domain with the restriction that a boundary triangulation of the domain surface does not admit coplanar adjacent faces.

### 3 Overview of the algorithm

Our method is based on constructing an approximation of the zero-contour of the signed-distance function defined by the (unknown) domain. We estimate the signed-distance and construct a trivariate approximant of it. The approximant is piecewise polynomial, and is defined over a tetrahedralization of the space that is incrementally and adaptively refined until the error conditions are satisfied.

The algorithm consists of the following three phases:

1. **Build an approximation of the signed-distance function (see Section 4).** The signed distance of a point  $p$  from a closed, orientable manifold  $D$ , is defined as the Euclidean distance from  $p$  to the closest point on  $D$ , with a positive sign if  $p$  lies outside  $D$ , and a negative sign otherwise.

We preprocess the data to define an approximate signed distance  $\delta(p)$ , and then approximate  $\delta(p)$  by a piecewise polynomial. This step can be seen as transforming the problem from a surface-data reconstruction to a volume-data approximation. Notice that this, and it requires a consistent reconstruction of the surface orientation at each point (see also [14, 15]). We have adopted a technique based on  $\alpha$ -shapes [27], to define the signed-distance function  $\delta(p)$ . Alternative approaches are described in [36, 34].

2. **Approximate the signed-distance (and field) by piecewise polynomial functions (see Section 5).** Build, in an adaptive fashion, a piecewise polynomial approximation  $f^D(x, y, z)$  of  $\delta(p)$ . The piecewise polynomial is built by least squares fitting of trivariate polynomials, in a 3D triangulation of a domain containing  $P$ , to the data points within each tetrahedron and to additional samples of the signed-distance function  $\delta$  defined in phase 1 above. If the error-of-fit in a tetrahedron exceeds the given bounds, then the triangulation is locally refined and the process is repeated in each new tetrahedron. The reconstructed domain is implicitly defined as  $f^D(x, y, z) = 0$ .

Concurrently to the approximation of the signed-distance function, a piecewise polynomial approximation of the scalar field can be computed in a similar fashion, by least squares fitting of the scalar field data.

Barycentric Bernstein-Bézier algebraic patches of degree three are used for the data fitting.

3. **Make the reconstructed surface  $C^1$ -smooth (see Section 6).** The surface is smoothed by applying normal averaging and a three-dimensional Clough-Tocher type of split.

Our approach has the following characteristics:

1. **Unrestricted topological genus.** Our method can reconstruct arbitrary objects without prior knowledge of their genus.
2. **Approximation of the data.** Since the data set is noisy and usually very dense, attempting to interpolate all data points would lead to inefficiency and to an unnecessarily large number of patches. We approximate the data within user-defined bounds.

3. **Adaptiveness.** Many objects have localized, small-scale features and large, flat or constant-curvature areas. Therefore it is convenient to be able to use patches of different size in different areas of the object’s surface. Our algorithm can approximate large dense data sets with a relatively small number of patches.
4. **Tangent-plane continuity.** Often smooth objects can be modeled quite accurately with surface patches joining so that first-order derivatives across the common boundary are continuous. The present approach cannot automatically handle objects with *mixed* continuity (i.e., objects whose surface is formed by smooth regions that join at a crisp edge or corner). This will be the subject of future investigation (see [16, 15]).
5. **Reconstruction of a field over the surface.** Our algorithm can reconstruct a  $C^1$ -continuous function, defined over the object’s surface, that approximates a scalar field sampled at the data points.

## 4 Phase 1: Preprocessing and the Signed-Distance Function

The first step of our algorithm consists in preprocessing the data points so that an approximate signed-distance is computable for any given query point  $q$ . Answering the question of whether a point lies inside or outside the domain surface requires a globally correct orientation of the surface. This problem raises fundamental questions on the possibility of reconstructing a shape from a set of points: What are sufficient conditions on the sampling to guarantee a faithful reconstruction of the shape? Is it possible to avoid ambiguities in interpreting the data? What additional informations (e.g. topological genus, smoothness, features) are needed? In our algorithm, we use  $\alpha$ -solids (a regularized version of  $\alpha$ -shapes [27]) to build an initial, piecewise-linear approximations of the unknown shape. This approximation will be used to classify points as either interior or exterior (i.e., to decide a sign for the signed distance function) as well as for computing an approximate distance of a point from the surface. The associated data structures will also be helpful in achieving efficiency in several required geometric algorithms.

The  $\alpha$ -shape of a set of points is a subcomplex of its Delaunay triangulation. Basically the  $\alpha$ -shape, for a suitable value of the parameter  $\alpha$ , contains only edges, triangles and tetrahedra that connect points that are close to each other. Long edges and large triangles and tetrahedra have a large associated size, so they become part of the  $\alpha$ -shape only for large values of  $\alpha$ . The  $\alpha$ -solid is the subset of tetrahedra contained inside connected shells of  $\alpha$ -shape triangles (more details in Section 4.4).

When the sampling is suitably dense, it is not difficult to find a value of  $\alpha$  such that the corresponding  $\alpha$ -solid closely approximates the boundary of the object. In fact, sampling density conditions that guarantee a homeomorphic and error bounded reconstruction can be formally proved [14]. Also, weighted points can be used to take into account a non-uniform sampling of the unknown manifold. We have devised an effective scheme for the automatic selection of an optimal  $\alpha$ -value, the extraction of the correspondent  $\alpha$ -solid, and a heuristic to locally improve the  $\alpha$ -solid w.r.t. to the sample points [15].

Before describing the actual signed-distance computation, we briefly review some concepts and results from Computational Geometry used in the algorithm. The style of this presentation is informal. The reader can refer to the papers in the references for more details.

### 4.1 Delaunay Triangulations

Given a set  $P$  of points in  $\mathbf{R}^3$  a tetrahedralization  $\mathcal{T}$  of the convex hull of  $P$ , that is, a partition of  $conv(P)$  into tetrahedra, can be built in such a way that the circumscribing sphere of each tetrahedron  $\tau$  does not contain any other point of  $P$  than its vertices. Such a tetrahedralization is called a (three-dimensional) Delaunay triangulation and, under nondegeneracy assumptions (no three points on a line, etc.) it is unique. Many different techniques have been proposed for the computation of Delaunay triangulations (see [25, 44]). For our purposes, an incremental approach is particularly well suited, as it can be used in both the preprocessing phase and the incremental refining of the adaptive triangulation used as support for the piecewise polynomials (see Section 5).

The algorithm we use is the randomized, incremental, flipping-based algorithm described in [28]. At the beginning the triangulation is initialized as a single tetrahedron, with vertices “at infinity”, that contains all points of  $P$ . At each step a point from  $P$  is inserted as a new vertex in the triangulation, the tetrahedron in which  $p$  lies is split into four tetrahedra and the Delaunay property is reestablished by “flipping” tetrahedra. This algorithm uses a data structure, called the history DAG, that maintains the collection of discarded tetrahedra. The DAG is used to locate the tetrahedron in which the point to be inserted lies. When a tetrahedron is split or groups of tetrahedra are flipped, they become

internal nodes of the DAG while the newly created tetrahedra become their children in the DAG. To locate a point, one starts at the root of the DAG (the single tetrahedron of the initial triangulation) and follows links down to a leaf.

The Delaunay triangulation of a set of  $n$  points in  $\mathbf{R}^3$  can be built in  $O(n \log n + n^2)$  expected time. The second term in this expression is of the same order as the maximum number of possible simplices. In practice, the running time of the algorithm is usually better than this theoretic bound.

## 4.2 Voronoi diagrams

Voronoi diagrams [3] are related to Delaunay triangulations by duality. A Voronoi diagram is a partition of the space into convex cells. There is a cell for each point  $p \in P$ , and the cell of a point  $p$  is the set of points that are closer to  $p$  than to any other point of  $P$ . So, all that has to be done to answer a closest-point query is to locate the cell the query point lies in. Efficient point-location data structures can be built on top of the Voronoi diagram. Using the randomized approach described in [19], the point-location data-structure (called *RPO-tree*, for Randomized Post Office tree) is built on top of the Voronoi diagram in  $O(n^{2+\epsilon})$  expected time, for any fixed  $\epsilon > 0$ , and then the closest-point query is able to be answered in  $O(\log n)$  expected time. The data structure requires  $O(n^{2+\epsilon})$  space in the worst case.

In our current implementation, we use a simpler hierarchical space subdivision technique to speed-up closest point queries, and we have found this approach to be efficient in practice, although theoretically suboptimal in the worst case.

## 4.3 Alpha-shapes

Given the Delaunay triangulation  $\mathcal{T}$  of a point set  $P$ , one can assign to each simplex  $\sigma \in \mathcal{T}$  (vertices, edges, triangles and tetrahedra) a *size* defined in the following way. Let  $\Theta_\sigma$  be the smallest ball whose boundary contains all vertices of  $\sigma$ . Then the *size* of  $\sigma$  will be defined to be equal to the radius of  $\Theta_\sigma$ , and  $\sigma$  will be said to be *conflict-free* if  $\Theta_\sigma$  does not contain any point of  $P$  other than the vertices of  $\sigma$ .

The subcomplex  $\Sigma_\alpha$  of simplices  $\sigma \in \mathcal{T}$  with either one of the following properties:

- (a) The size of  $\sigma$  is less than  $\alpha$  and  $\sigma$  is conflict-free, or
- (b)  $\sigma$  is a face of  $\tau$  and  $\tau \in \Sigma_\alpha$ ,

is called the  $\alpha$ -shape of  $P$ .  $\alpha$ -Shapes [27] have been extended to higher dimensions and to weighted sets of points [26].  $\alpha$ -Shapes can be intuitively thought of as the subcomplex of  $\mathcal{T}$  obtained in the following way: imagine that a ball-shaped eraser of radius  $\alpha$  is moved in the space, assuming all possible positions such that no point of  $P$  lies inside the eraser. The eraser removes all simplices it can pass through, but not those whose size is smaller than  $\alpha$ . The remaining simplices (together with all their faces) form the  $\alpha$ -shape for that value of the parameter  $\alpha$  (see Figure 4.1). Notice that there exists only a finite number of different  $\alpha$ -shapes. The collection of all possible  $\alpha$ -shapes of  $P$  is called the *family*

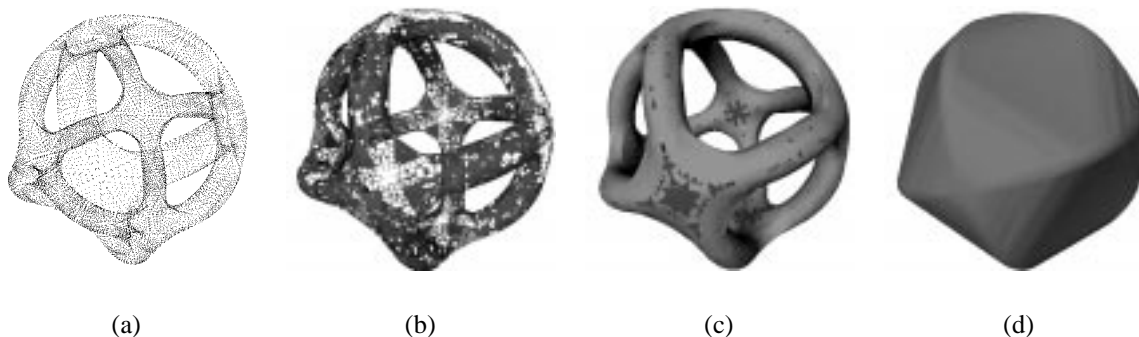


FIGURE 4.1: (a) Data points (a random sampling of a CAD object). (b), (c) and (d) Three-dimensional  $\alpha$ -shapes for increasing values of the parameter  $\alpha$ . Edges, triangles and tetrahedra are shown in different shades. (c) corresponds to the  $\alpha$ -value selected by our algorithm.

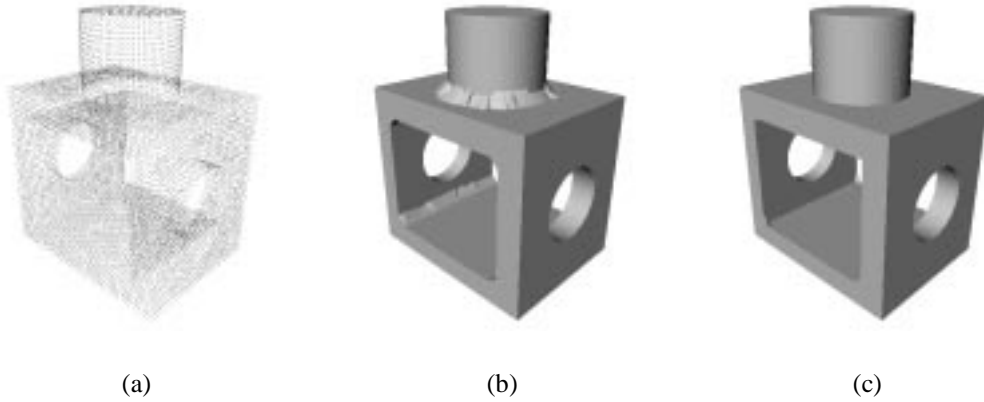


FIGURE 4.2: (a) Data points (a scattered sampling of a CAD object). (b) Automatically selected  $\alpha$ -solid. (c)  $\alpha$ -solid after heuristic improvement.

of  $\alpha$ -shapes of  $P$ , and can be computed in time proportional to the number of simplices in  $\mathcal{T}$ . We use the  $\alpha$ -shape computation for generating an initial piecewise-linear approximation of the domain surface  $D$ .

#### 4.4 Alpha-solids

$\alpha$ -shapes allow us to reason about the “shape” of an unorganized set of points in a formal framework. However, they are not completely well-suited to our purpose, as an  $\alpha$ -shape is in general a non-connected polytope of mixed dimensionality (it can contain tetrahedra as well as triangles, edges and isolated points). We also need to find a value of the parameter  $\alpha$  such that the corresponding  $\alpha$ -shape  $\Sigma_\alpha$  is a good approximation of the sampled domain  $D$ . We define an  $\alpha$ -solid as follows: It is the subset of tetrahedra  $\tau$  in  $\mathcal{T}$  that either belong to the  $\alpha$ -shape, or such that they are contained in a continuous shell of triangles belonging to the  $\alpha$ -shape. In practice, it is quite easy to compute this “regularized” version of the  $\alpha$ -shape: One does a breadth first search on the dual graph of  $\mathcal{T}$ , starting with a tetrahedron  $\sigma$  that is known to be external (e.g. one that has a vertex at infinity) and continuing with adjacent tetrahedra. When an adjacent tetrahedron  $\tau$  (or the face in common between  $\sigma$  and  $\tau$ ) belongs to  $\Sigma_\alpha$ ,  $\tau$  is marked as internal (negative sign) and not enqueued. Other tetrahedra are marked as external (positive sign) and put in a queue for further processing. In this way, the positive sign “propagates” to all tetrahedra that can be reached from infinity without traversing a triangle belonging to the  $\alpha$ -shape. The set of negative tetrahedra constitutes the  $\alpha$ -solid.

Obviously, for varying values of  $\alpha$ , one obtains  $\alpha$ -solids that range from the empty set to the convex hull of  $P$ . These solids are ordered with respect to  $\alpha$ , and form a finite family whose cardinality is polynomial in  $\|P\|$ . We can therefore perform a binary search on possible values of  $\alpha$ , looking for the minimum value of  $\alpha$  such that the corresponding  $\alpha$ -solid is connected, and such that all the points of  $P$  either lie on its boundary or in its interior. Notice that we use a condition weaker than requiring all the points of  $P$  to lie on the boundary of the  $\alpha$ -solid, as in practice this stronger condition is difficult to satisfy, especially in the vicinity of concave sharp features. We instead rely on a simple postprocessing stage, which incrementally removes tetrahedra from the initial  $\alpha$ -solid, and produces a better approximation as output. More details on this process can be found in [15]. Figure 4.2 shows an automatically selected  $\alpha$ -solid, and the same  $\alpha$ -solid after improvement.

#### 4.5 Signed Distance Computation

In the preprocessing phase the Delaunay triangulation  $\mathcal{T}$  of the set of input points  $P$  is computed, and then the Voronoi diagram and the family of  $\alpha$ -shapes of  $P$  are constructed. During the process, the history DAG (and, optionally, the RPO-tree) is built to allow a fast location of the tetrahedron (and Voronoi cell) a query point  $q$  lies in. Note that all these data structures are intimately related.

We then perform a binary search to select the best approximating  $\alpha$ -solid, apply the improvement heuristic, and classify all tetrahedra of  $\mathcal{T}$  as either positive (outside the  $\alpha$ -solid) or negative (inside the  $\alpha$ -solid).

After this preprocessing is done, we can compute the signed distance of a point  $q$  w.r.t. the  $\alpha$ -solid as follows:

1. Use the history DAG to locate the tetrahedron  $\tau$  in which the point  $q$  lies.  $\tau$  has been marked as either positive or negative by the preprocessing described above, and its sign is returned.
2. Find the face (a triangle  $\sigma$ ) belonging to the boundary of the  $\alpha$ -solid that is closest to  $q$ . Compute the distance between  $q$  and  $\sigma$ , and return it with the sign computed in step 1. We currently use a simple space-partitioning data structure to find the closest face to a given query point. Alternatively, one can use the RPO-tree data structure to find the closest point  $p$  in  $P$ , and return  $\|p - q\|$  (again, with sign) as an approximate distance.

## 5 Phase 2: Incremental Refinement and Approximation

In Phase 2 of the algorithm a three-dimensional Delaunay triangulation  $\mathcal{T}$  is initialized and incrementally refined, and piecewise-polynomial functions  $f^D$  and  $f^F$  are generated. For each tetrahedron  $\tau \in \mathcal{T}$  we compute two Bernstein-Bézier trivariate polynomials  $f_\tau^D$  and  $f_\tau^F$ , to approximate the part of domain surface and scalar field contained in  $\tau$ , respectively. The coefficients of the polynomials are computed using the signed-distance function described in Section 4.5, the sample points and the scalar field values.

After computing the two polynomials, the approximation errors are estimated and, if one or both the errors are too large, the current triangulation  $\mathcal{T}$  is refined until the errors are within the given bounds. The triangulation refinement is done by adding at each step a new point to split the tetrahedron with the maximum error, and using the incremental Delaunay triangulation algorithm to update the triangulation.

Before describing in further details the computation of the approximating functions, we recall some facts and terminology related to Bernstein-Bézier trivariate forms.

### 5.1 Bernstein-Bézier forms

Let  $p_1, p_2, p_3, p_4 \in \mathbf{R}^3$  be affine independent. Then the tetrahedron  $\tau$  with vertices  $p_1, p_2, p_3, p_4$ , is  $\tau = [p_1 p_2 p_3 p_4]$ . For any  $p = \sum_{i=1}^4 \alpha_i p_i$ ,  $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ ,  $\sum_{i=1}^4 \alpha_i = 1$  are the barycentric coordinates of  $p$ . Let  $p = (x, y, z)^T$ ,  $p_i = (x_i, y_i, z_i)^T$ . The barycentric coordinates relate to the Cartesian coordinates via the following relation:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \quad (5.1)$$

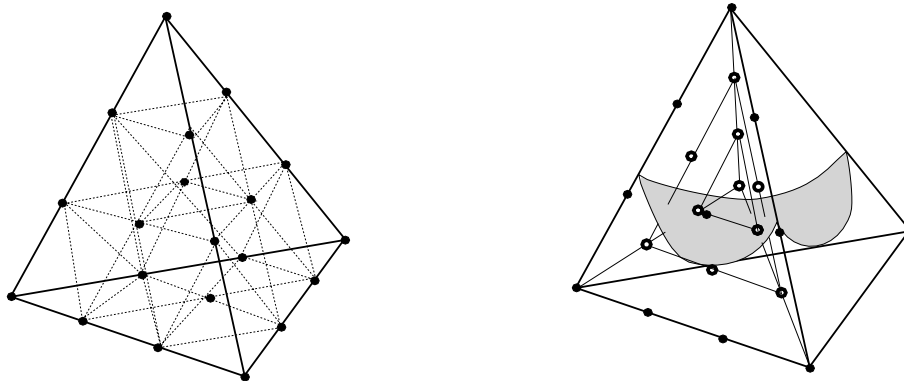


FIGURE 5.1: The splitting of a tetrahedron (left) into four sub-tetrahedra (right). Only the weights of one of the resulting sub-tetrahedra are shown.

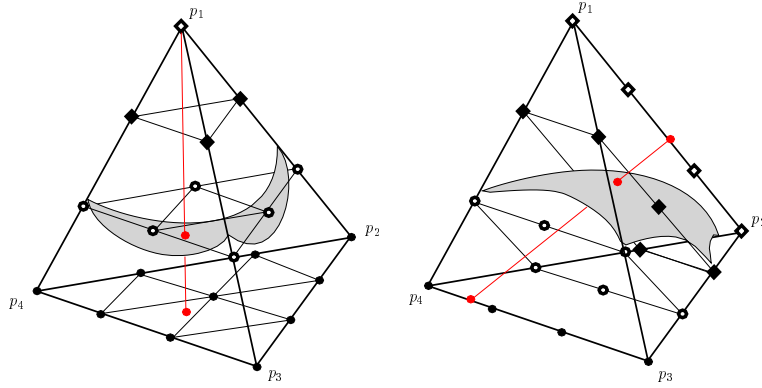


FIGURE 5.2: The layers of Bézier ordinates in a tetrahedron. (left) Three-sided patch. (right) Four-sided patch.

Any polynomial  $f(p)$  of degree  $n$  can be expressed as a Bernstein-Bézier (BB) form over  $\tau$  as

$$f(p) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha), \quad \lambda \in \mathcal{Z}_+^4$$

where

$$B_\lambda^n(\alpha) = \frac{n!}{\lambda_1! \lambda_2! \lambda_3! \lambda_4!} \alpha_1^{\lambda_1} \alpha_2^{\lambda_2} \alpha_3^{\lambda_3} \alpha_4^{\lambda_4}$$

is a Bernstein polynomial,  $|\lambda| = \sum_{i=1}^4 \lambda_i$  with  $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$ ,  $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$  are the barycentric coordinates of  $p$  w.r.t.  $\tau$ ,  $b_\lambda = b_{\lambda_1 \lambda_2 \lambda_3 \lambda_4}$  (as a subscript, we simply write  $\lambda_1 \lambda_2 \lambda_3 \lambda_4$  for  $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$ ) are called Bézier ordinates, and  $\mathcal{Z}_+^4$  stands for the set of all four dimensional vectors with non-negative integer components.

The points

$$p_\lambda = \frac{\lambda_1}{n} p_1 + \frac{\lambda_2}{n} p_2 + \frac{\lambda_3}{n} p_3 + \frac{\lambda_4}{n} p_4, \quad |\lambda| = n$$

are called the *regular points* of  $\tau$ . The points  $(p_\lambda, b_\lambda) \in \mathbf{R}^4$  are called *Bézier points*, and the regular lattice of lines connecting them *Bézier net*.

The following lemma gives necessary and sufficient conditions for continuity between adjacent polynomial patches:

**Lemma 5.1** (see [29]). *Let  $f(p) = \sum_{|\lambda|=n} a_\lambda B_\lambda^n(\alpha)$  and  $g(p) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha)$  be two polynomials defined on the tetrahedra  $[p_1 p_2 p_3 p_4]$  and  $[p'_1 p_2 p_3 p_4]$ , respectively. Then*

(i)  *$f$  and  $g$  are  $C^0$  continuous at the common face  $[p_2 p_3 p_4]$  if and only if*

$$a_\lambda = b_\lambda, \quad \text{for all } \lambda = 0 \lambda_2 \lambda_3 \lambda_4, \quad |\lambda| = n \quad (5.2)$$

(ii)  *$f$  and  $g$  are  $C^1$  continuous at the common face  $[p_2 p_3 p_4]$  if and only if (5.2) holds and, for all  $\lambda = 0 \lambda_2 \lambda_3 \lambda_4$ ,  $|\lambda| = n - 1$ ,*

$$b_{\lambda+e_1} = \beta_1 a_{\lambda+e_1} + \beta_2 a_{\lambda+e_2} + \beta_3 a_{\lambda+e_3} + \beta_4 a_{\lambda+e_4} \quad (5.3)$$

where  $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)^T$  are the barycentric coordinates of  $p'_1$  with respect to  $[p_1 p_2 p_3 p_4]$ , defined by the following relation

$$p'_1 = \beta_1 p_1 + \beta_2 p_2 + \beta_3 p_3 + \beta_4 p_4, \quad |\beta| = 1$$

Relation (5.3) is called the coplanar condition.

A-patches are guaranteed to be single-sheeted when the conditions described in two following lemmas are satisfied. In particular, A-patches can be classified as *three-sided* when a segment connecting a vertex of the tetrahedron to a point on the opposite face intersects the patch at most once, and *four-sided* when the same property holds for a segment connecting two points on two opposite edges.

**Lemma 5.2** Let  $\tau = [p_1 p_2 p_3 p_4]$ . The regular points of  $\tau$  can be thought of as organized in triangular layers, that we can number from 0 to  $n$  going from  $p_1$  to the opposite face  $[p_2 p_3 p_4]$  (see Figure 5.2). If the Bézier ordinates are all positive (negative) on layers  $0, \dots, k-1$  and all negative (positive) on layers  $k+1, \dots, n$  ( $0 < k < n$ ), then the patch is single-sheeted (i.e., any line through  $p_1$  and  $p \in [p_2 p_3 p_4]$  intersects the patch only once).

**Lemma 5.3** Let  $\tau = [p_1 p_2 p_3 p_4]$ . The regular points of  $\tau$  can be thought of as organized in quadrilateral layers, that we can number from 0 to  $n$  going from edge  $[p_1 p_2]$  to the opposite edge  $[p_3 p_4]$  (see Figure 5.2). If the Bézier ordinates are all positive (negative) on layers  $0, \dots, k-1$  and all negative (positive) on layers  $k+1, \dots, n$  ( $0 < k < n$ ), then the patch is single-sheeted (i.e., any line through  $p \in [p_1 p_2]$  and  $q \in [p_3 p_4]$  intersects the patch only once).

In the Lemmas above, the Bézier ordinates on layer  $k$  can have any sign. Patches satisfying the conditions of Lemma 5.2 will be called *three-sided*; those satisfying the conditions of Lemma 5.3 will be called *four-sided*. See [7] for proofs and further details.

The following two lemmas are used in Section 6.

**Lemma 5.4** (see [31]). If  $f(p) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha)$ , then

$$b_{(n-1)e_i + e_j} = b_{n e_i} + \frac{1}{n} (p_j - p_i)^T \nabla f(p_i),$$

for  $i, j = 1, 2, 3, 4$ ;  $j \neq i$ , where

$$e_i = (\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4), \epsilon_i = 1, \epsilon_j = 0, j \neq i.$$

**Lemma 5.5** Let  $f(p) = F(\alpha) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha)$  where  $\alpha$  are the barycentric coordinates of  $p$ . For any given points  $p^{(1)}$  and  $p^{(2)}$ , let  $\alpha^{(1)}$  and  $\alpha^{(2)}$  be their barycentric coordinates. Then

$$\begin{aligned} & \nabla f(p)^T (p^{(1)} - p^{(2)}) \\ &= \nabla F(\alpha)^T (\alpha^{(1)} - \alpha^{(2)}) \\ &= n \sum_{|\lambda|=n-1} b_\lambda^1 (\alpha^{(1)} - \alpha^{(2)}) B_\lambda^{n-1}(\alpha) \end{aligned}$$

where

$$b_\lambda^1 (\alpha^{(1)} - \alpha^{(2)}) = \sum_{|\mu|=1} b_{\lambda+\mu} B_\mu^1 (\alpha^{(1)} - \alpha^{(2)})$$

The first equality of the lemma can be easily proved applying equation (5.1). The second equality can be found in [29].

## 5.2 Outline of the Incremental Approximation Phase

We are now ready to present in details the steps required to compute the approximant functions  $f^D$  and  $f^F$ .

1. Build an initial bounding tetrahedron  $\tau$ , such that  $P \subset \tau$ . Set  $\mathcal{T} = \{\tau\}$ . Mark  $\tau$  as *new*.
2. For each *new* tetrahedron  $\tau \in \mathcal{T}$ , compute the signed-distance at all its regular points  $p_\lambda$ . If the values of  $\delta(p_\lambda)$ ,  $|\delta| = n$ , do not satisfy either Lemma 5.2 or Lemma 5.3, then set domain and field errors  $\vartheta_\tau^D = \vartheta_\tau^F = \infty$ . Otherwise, compute local approximants  $f_\tau^D$  and  $f_\tau^F$  for the domain surface  $D$  and scalar field  $F$  as follows:

$$f_\tau^D(p) = \sum_{|\lambda|=n} b_\lambda^D B_\lambda^n(\alpha), \quad (5.4)$$

$$f_\tau^F(p) = \sum_{|\lambda|=n} b_\lambda^F B_\lambda^n(\alpha). \quad (5.5)$$

The coefficients  $b_\lambda^D$  are computed by first interpolating the computed values of the signed-distance function:

$$f_\tau^D(p_\lambda) = \delta(p_\lambda), \quad |\lambda| = n \quad (5.6)$$

The tetrahedron  $\tau$  is then split into four sub-tetrahedra  $\tau_1, \dots, \tau_4$  (see Figure 5.1) by joining the barycenter of  $\tau$  with its four vertices ( $\tau_k$  is the sub-tetrahedron opposite to vertex  $p_k$ ). The regular points on the faces of the sub-tetrahedra coincide with those of the original tetrahedron  $\tau$ . For these points we use the coefficients computed from (5.6). Notice that on the shared face of two adjacent tetrahedra these coefficients will coincide, as  $f^D$ , restricted to that face, interpolates the signed distance at a number of points equal to the number of its coefficients. All interior coefficients (a total of 15) of the sub-tetrahedra are computed by solving the least squares problem

$$\begin{cases} f_{\tau_k}^D(p_i) = 0, & p_i \in P \cap \tau_k, k = 1, \dots, 4 \\ f_{\tau_k}^D(p_\lambda) = \delta(p_\lambda), & |\lambda| = n, \lambda_k \neq 0, k = 1, \dots, 4 \end{cases} \quad (5.7)$$

where we use the values of the signed-distance at regular points (of each sub-tetrahedron  $\tau_k$ ) in addition to the data points contained in  $\tau$ . The signed-distance data helps in avoiding multiple sheets in the approximating patch.

For the scalar field approximant we compute a least squares approximation of the field values at data points within  $\tau$ :

$$f_\tau^F(p_i) = v_i, \quad p_i \in P \cap \tau \quad (5.8)$$

Notice that the field approximant is not globally continuous. Continuity will be achieved by averaging and interpolating values of the approximant at the vertices of  $\mathcal{T}$  in a subsequent phase, described in Section 6.

3. If the coefficients computed in the step above do not satisfy the conditions of either Lemma 5.2 or Lemma 5.3, set  $\vartheta_\tau^D = \vartheta_\tau^F = \infty$ . Otherwise, compute the approximation error for both functions:

$$\vartheta_\tau^D = \frac{\sqrt{\sum_{k=1}^4 \sum_{p_i \in P \cap \tau_k} f_{\tau_k}^D(p_i)^2}}{\|P \cap \tau\|}$$

$$\vartheta_\tau^F = \frac{\sqrt{\sum_{p_i \in P \cap \tau} (f_\tau^F(p_i) - v_i)^2}}{\|P \cap \tau\|}$$

(if  $\tau \cap P = \emptyset$ , then set  $\vartheta_\tau^D = 0$  and  $\vartheta_\tau^F = 0$ ), and keep track of the following two quantities:

$$\vartheta_{\sigma'}^D = \max_{\tau \in \mathcal{T}} \{\vartheta_\tau^D\} \quad (5.9)$$

$$\vartheta_{\sigma''}^F = \max_{\tau \in \mathcal{T}} \{\vartheta_\tau^F\} \quad (5.10)$$

4. If both  $\vartheta_{\sigma'}^D < \varepsilon^D$  and  $\vartheta_{\sigma''}^F < \varepsilon^F$  then the algorithm stops the incremental refinement phase and begins the smoothing phase. Otherwise, either  $\sigma'$  or  $\sigma''$  is selected for further refinement (according to a user-definable strategy, e.g., always choose  $\sigma'$  first, assigning priority to the surface, or choose the one with the largest error. The circumcenter  $q$  of the selected tetrahedron is computed and added to the set of vertices of the triangulation,  $q$  is inserted in  $\mathcal{T}$  and  $\mathcal{T}$  is updated with splits and flippings to accommodate the new vertex and restore the Delaunay property (adding the center of the circumscribing sphere generally yields good aspect ratio tetrahedra in the final triangulation [23]). At the same time the subset  $P \cap \tau$  of points that lie within each modified tetrahedron  $\tau$  is updated. This is done by considering the points originally within the modified simplex, and reclassifying them with respect to the splitting/flipping planes.

Then mark all split/flipped tetrahedra as *old* and all newly created ones as *new* and go back to step 2.

## 6 Phase 3: Smoothing

The functions  $f^D(p)$  and  $f^F(p)$  computed in phase 2 of the algorithm are not  $C^1$  continuous. To achieve  $C^1$  continuity, we apply a subdivision scheme to the tetrahedra of  $\mathcal{T}$ , and compute  $C^1$ -smooth Bernstein-Bézier patches on the refined triangulation.

We base our trivariate scheme on the  $n$ -dimensional Clough-Tocher scheme given by Worsey and Farin [49, 29]. In this scheme, for each vertex in the original triangulation an average of the values of the functions  $f^D$  and  $f^F$  and

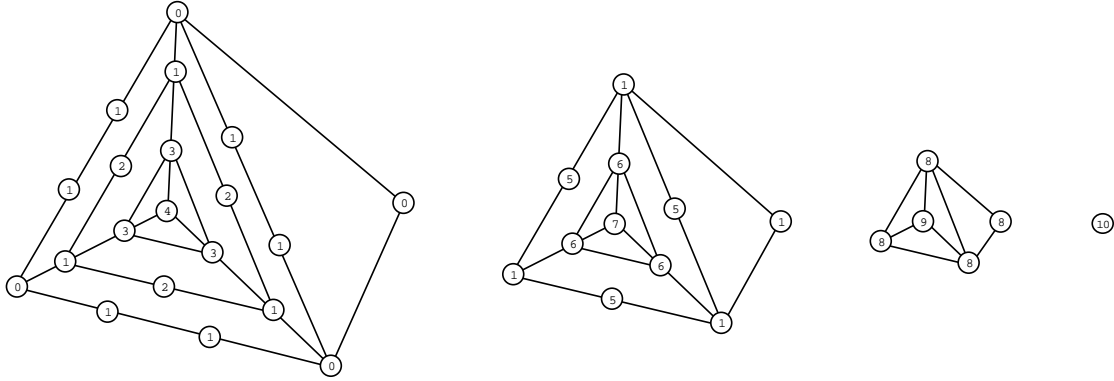


FIGURE 6.1: Clough-Tocher 12-way split. From left to right: Cubic, quadratic, linear and constant shells.

their gradients, for all incident patches is computed. The surface approximant is already  $C^0$ , so only the gradient needs to be averaged. In addition, the average gradient at the middle point of each edge is computed. Each tetrahedron is then split into 12 subtetrahedra by inserting the incenter of each tetrahedron and a point on each face (the point on the face shared by two adjacent tetrahedra must be collinear with their incenters [49]), and joining these points with the original vertices. A cubic trivariate polynomial is built on each subtetrahedron. The coefficients of the 12 resulting patches are computed based on the value of the function at each vertex, the average gradient at vertices and midedge points, and the continuity constraints. The resulting patches are  $C^1$  and interpolate the averaged values and gradient of the functions.

We call the outer faces of the tetrahedron a cubic shell. If we peel off the cubic shell, we are left with a quadratic shell. If we repeat this peeling what remains is a linear shell and at the end a center point (see Figure 6.1). The coefficients of the cubic trivariate interpolant are determined in this manner, from the outer shell to the center. Coefficients numbered 0 are set equal to the value of the function at the corresponding vertex. Coefficients number 1 are computed from the value of the function and of the gradient at the adjacent number 0 vertex, using Lemma 5.4. Coefficients number 2 are computed from the average gradient at the middle point of the corresponding edge, using Lemma 5.5. All other coefficients are computed using Lemma 5.1 and the continuity constraint.

Another trivariate Clough-Tocher scheme (see [1]) splits each tetrahedron into four subtetrahedra. However the interpolants in each subtetrahedron are now of quintic degree and furthermore require  $C^2$  data at the vertices of the main tetrahedron. Since our data at the vertices of the tetrahedral mesh comes from the averaging of locally computed low-degree approximants, the higher-order derivatives tend to be unreliable in general. We therefore prefer to use the lower degree cubic scheme that uses only first order derivatives at the vertices. An alternative approach to build a  $C^1$  interpolant with cubic patches is described in [9].

## 7 Implementation and Results

Some results are summarized in Table 7.1, and the reconstructed models are shown in Figures 7.1, 7.2 and 7.3. Figures 7.3 (c) and 7.3 (d) also show the reconstructed scalar field. We can visualize the graph of the reconstructed function  $f^F$  on the domain surface  $f^D$  either by projecting its iso-contours onto the surface  $f^D$ , or by directly displaying the surface graph of the function  $f^F$  as a height map over the domain. Such visualization schemes based on central or normal projections are well known and details can be found for example in [39, 43, 9].

We have implemented all steps of the reconstruction algorithm in C++, using Open Inventor for visualization, and X/Motif for the graphical interface. The front-end of the reconstruction toolkit is shown in Figure 7.4.

## References

- [1] ALFELD, P. A trivariate Clough-Tocher scheme for tetrahedral data. *Computer Aided Geometric Design* 1 (1984), 169–181.

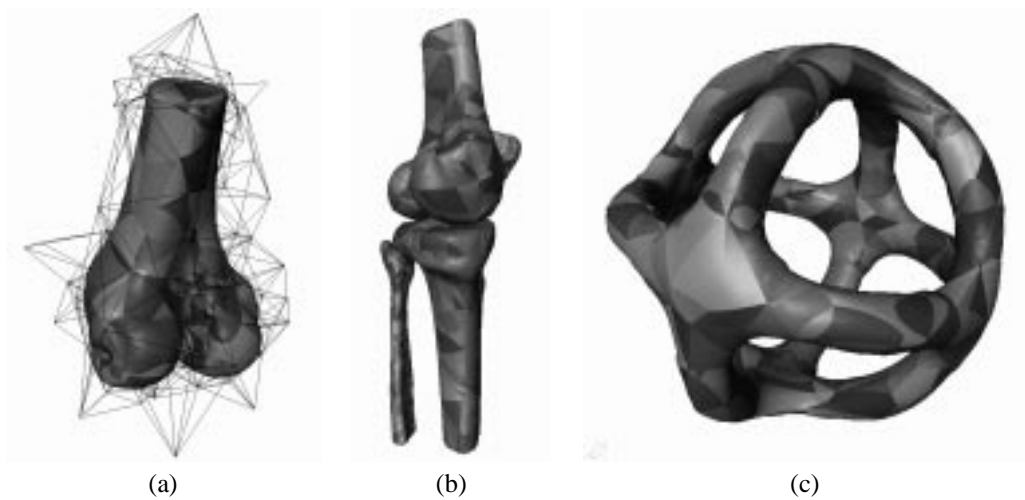


FIGURE 7.1: (a) and (b) Reconstruction from points extracted via marching cubes from CT data (CT scan courtesy of the Visible Human Project, National Library of Medicine). (a) Reconstruction of the lower part of the femur. The wireframe shows the tetrahedral mesh that acts as support for the A-patches. Different shades are used to highlight the subdivision of the surface into patches. (b) Reconstruction of the four knee bones. (c) Reconstruction of a high-genus object (data courtesy of Jörg Peters, Purdue University).

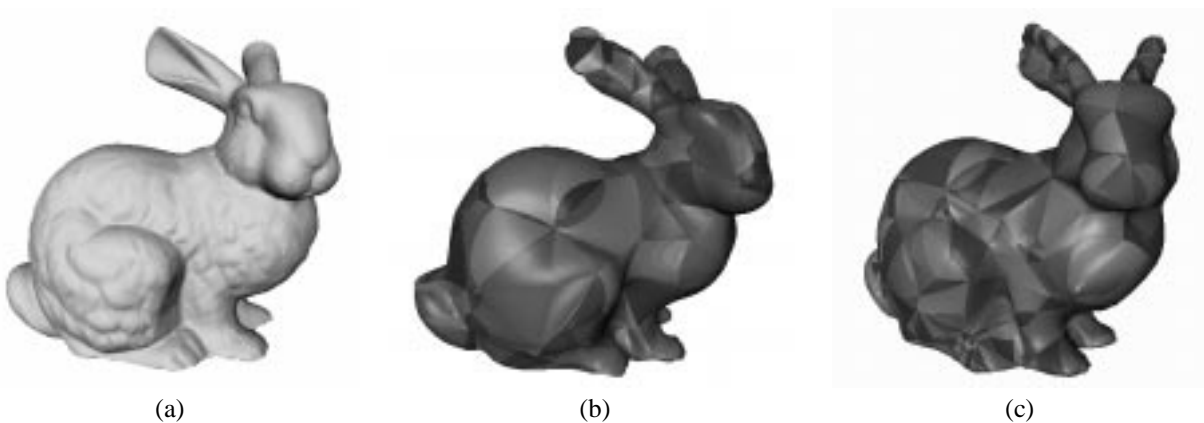


FIGURE 7.2: Reconstruction from range data. (a) Automatically selected  $\alpha$ -solid. (b) Smooth reconstruction after 100 refinement steps (233 patches) (c) Smooth reconstruction after 500 refinement steps (735 patches). Data courtesy of Stanford University Computer Graphics Laboratory.

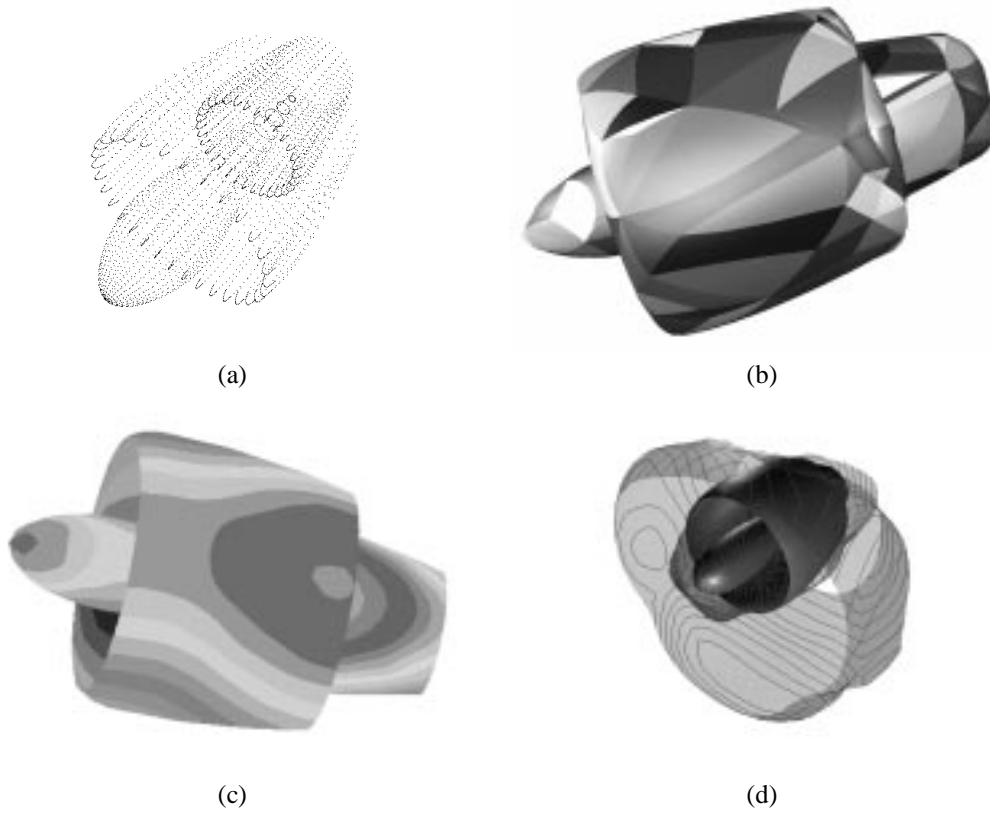


FIGURE 7.3: Visualization of a reconstructed jet engine and an associated scalar field: (a) Data points. (b) Reconstructed domain. (c) Iso-pressure contours of a pressure field displayed on the surface of the jet engine. (d) The reconstructed engine surface and visualization of the pressure field on the jet engine using the normal projection method.

<i>Object</i>	<i>Number of Points</i>	<i>Alpha-Solid Time</i>	<i>Number of Triangles</i>	<i>Fitting Time</i>	<i>Number of Patches</i>	<i>Error</i>
Femur	9807	1.5	19610	19	672	< 1%
Tibia	9200	1.4	18396	17	516	< 1%
Fibula	8146	1.1	16288	17	536	< 1%
Patella	2050	0.3	4096	8.1	269	< 1%
3 Tori	10833	2.2	21692	25	812	< 1%
Jet Engine	9800	1.4	18121	16	382	< 1%
Bunny	15134	2.5	36545	31	535	< 1%

TABLE 7.1: Results of the reconstruction algorithm. The table shows for each object, from left to right: (1) The number of points in the sampling; (2) The time, in minutes, required by the  $\alpha$ -solid computation (including 3D Delaunay triangulation, computation of family of  $\alpha$ -shapes, automatic selection of  $\alpha$  value, improvement by local sculpturing). All computations were carried out on a SGI Indigo2, with a 250MHz MIPS 4400 CPU; (3) The number of triangles in the boundary of the  $\alpha$ -solid; (4) The time, in minutes, required by the fitting phase; (5) The number of patches in the smooth reconstruction; (6) The error, as percentage of the diameter of the object.

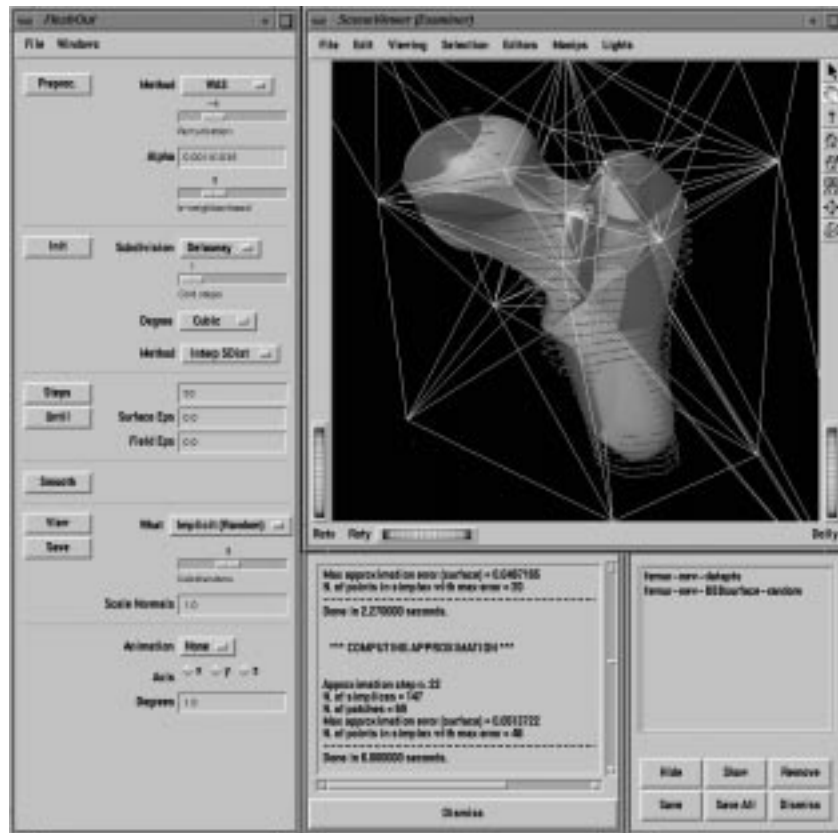


FIGURE 7.4: Graphical User Interface of the surface reconstruction application. The picture shows an intermediate step of the incremental approximation phase.

- [2] ALFELD, P. Scattered data interpolation in three or more variables. In *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. Schumaker, Eds. Academic Press, Boston, 1989, pp. 1–34.
- [3] AURENHAMMER, F. Voronoi diagrams: a survey of a fundamental geometric data structure. Report B-90-09, Fachber. Math., Free Univ. Berlin, Berlin, West Germany, 1990.
- [4] BAJAJ, C. The emergence of algebraic curves and surfaces in geometric design. In *Directions in Geometric Computing*, R. Martin, Ed. Information Geometers Press, 1993, pp. 1–29.
- [5] BAJAJ, C., BERNARDINI, F., AND XU, G. Adaptive reconstruction of surfaces and scalar fields from dense scattered trivariate data. Tech. Rep. CSD-TR-95-028, Department of Computer Sciences, Purdue University, Apr. 1995.
- [6] BAJAJ, C., BERNARDINI, F., AND XU, G. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Computer Graphics Proceedings (1995)*, Annual Conference Series. Proceedings of SIGGRAPH 95, pp. 109–118.
- [7] BAJAJ, C., CHEN, J., AND XU, G. Modeling with cubic A-patches. *ACM Trans Graph.* 14, 2 (1995), 103–133.
- [8] BAJAJ, C., AND IHM, I.  $C^1$  smoothing of polyhedra with implicit algebraic splines. *Computer Graphics* 26, 2 (July 1992), 79–88. Proceedings of SIGGRAPH 92.
- [9] BAJAJ, C., AND XU, G. Modeling scattered function data on curved surfaces. In *Fundamentals of Computer Graphics*, Z. T. J. Chen, N. Thalmann and D. Thalmann, Eds. World Scientific Publishing Co., Beijing, China, 1994, pp. 19–29.
- [10] BARNHILL, R. E. Surfaces in computer aided geometric design: A survey with new results. *Computer Aided Geometric Design* 2 (1985), 1–17.
- [11] BARNHILL, R. E., AND FOLEY, T. A. Methods for constructing surfaces on surfaces. In *Geometric Modeling: Methods and their Applications*, G. Farin, Ed. Springer, Berlin, 1991, pp. 1–15.
- [12] BARNHILL, R. E., OPITZ, K., AND POTTMANN, H. Fat surfaces: a trivariate approach to triangle-based interpolation on surfaces. *Computer Aided Geometric Design* 9 (1992), 365–378.
- [13] BARNHILL, R. E., PIPER, B. R., AND RESCORLA, K. L. Interpolation to arbitrary data on a surface. In *Geometric Modeling*, G. Farin, Ed. SIAM, Philadelphia, 1987, pp. 281–289.
- [14] BERNARDINI, F., AND BAJAJ, C. Sampling and reconstructing manifolds using alpha-shapes. Tech. Rep. CSD-TR-97-013, Department of Computer Sciences, Purdue University, 1997.
- [15] BERNARDINI, F., BAJAJ, C., CHEN, J., AND SCHIKORE, D. Automatic reconstruction of 3D CAD models from digital scans. Tech. Rep. CSD-TR-97-012, Department of Computer Sciences, Purdue University, 1997.
- [16] BERNARDINI, F., BAJAJ, C., CHEN, J., AND SCHIKORE, D. Triangulation-based 3D reconstruction methods. In *13th ACM Symposium on Computational Geometry (1997)*, 6th Annual Video Review of Computational Geometry, ACM. To appear.
- [17] BOISSONNAT, J.-D. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.* 3, 4 (1984), 266–286.
- [18] CHOI, B. K., SHIN, H. Y., YOON, Y. I., AND LEE, J. W. Triangulation of scattered data in 3D space. *Computer Aided Design* 20, 5 (June 1988), 239–248.
- [19] CLARKSON, K. L. A randomized algorithm for closest-point queries. *SIAM J. Comput.* 17 (1988), 830–847.
- [20] CURLESS, B., AND LEVOY, M. A volumetric method for building complex models from range images. In *Computer Graphics Proceedings (1996)*, Annual Conference Series. Proceedings of SIGGRAPH 96, pp. 303–312.
- [21] DAHMEN, W. Smooth piecewise quadratic surfaces. In *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. Schumaker, Eds. Academic Press, Boston, 1989, pp. 181–193.
- [22] DAHMEN, W., AND THAMM-SCHAAR, T.-M. Cubicoids: modeling and visualization. *Computer Aided Geometric Design* 10 (1993), 93–108.
- [23] DEY, T. K., BAJAJ, C. L., AND SUGIHARA, K. On good triangulations in three dimensions. *Internat. J. Comput. Geom. Appl.* 2, 1 (1992), 75–95.
- [24] ECK, M., AND HOPPE, H. Automatic reconstruction of B-splines surfaces of arbitrary topological type. In *Computer Graphics Proceedings (1996)*, Annual Conference Series. Proceedings of SIGGRAPH 96, pp. 325–334.
- [25] EDELSBRUNNER, H. *Algorithms in Combinatorial Geometry*, vol. 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.
- [26] EDELSBRUNNER, H. Weighted alpha shapes. Tech. Rep. UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois, Urbana-Champaign, IL, 1992.
- [27] EDELSBRUNNER, H., AND MÜCKE, E. P. Three-dimensional alpha shapes. *ACM Trans. Graph.* 13, 1 (Jan. 1994), 43–72.
- [28] EDELSBRUNNER, H., AND SHAH, N. R. Incremental topological flipping works for regular triangulations. *Algorithmica* 15 (1996), 223–241.

- [29] FARIN, G. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design* 3 (1986), 83–127.
- [30] FRANKE, R. Recent advances in the approximation of surfaces from scattered data. In *Multivariate Approximation*, C.K.Chui, L.L.Schumaker, and F.I.Utreras, Eds. Academic Press, New York, 1987, pp. 275–335.
- [31] GUO, B. Surface generation using implicit cubics. In *Scientific Visualization of Physical Phenomena*, N. M. Patrikalakis, Ed. Springer-Verlag, Tokyo, 1991, pp. 485–530.
- [32] GUO, B. Non-splitting macro patches for implicit cubic spline surfaces. *Computer Graphics Forum* 12, 3 (1993), 434–445.
- [33] HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JIN, H., McDONALD, J., SCHWITZER, J., AND STUELZLE, W. Piecewise smooth surface reconstruction. In *Computer Graphics Proceedings (1994)*, Annual Conference Series. Proceedings of SIGGRAPH 94, pp. 295–302.
- [34] HOPPE, H., DEROSE, T., DUCHAMP, T., McDONALD, J., AND STUELZLE, W. Surface reconstruction from unorganized points. *Computer Graphics* 26, 2 (July 1992), 71–78. Proceedings of SIGGRAPH 92.
- [35] KRISHNAMURTHY, V., AND LEVOY, M. Fitting smooth surfaces to dense polygonal meshes. In *Computer Graphics Proceedings (1996)*, Annual Conference Series. Proceedings of SIGGRAPH 96, pp. 313–324.
- [36] MOORE, D., AND WARREN, J. Approximation of dense scattered data using algebraic surfaces. In *Proceedings of the 24th annual Hawaii International Conference on System Sciences (1991)*, V. Milutinovic and B. D. Shriver, Eds., vol. 1.
- [37] NIELSON, G. M. Modeling and visualizing volumetric and surface-on-surface data. In *Focus on Scientific Visualization*, H. Hagen, H. Muller, and G. M. Nielson, Eds. Springer, 1992, pp. 219–274.
- [38] NIELSON, G. M. Scattered data modeling. *IEEE Computer Graphics & Applications* 13 (1993), 60–70.
- [39] NIELSON, G. M., FOLEY, T., LANE, D., FRANKE, R., AND HAGEN, H. Interpolation of scattered data on closed surfaces. *Computer Aided Geometric Design* 7, 4 (1990), 303–312.
- [40] NIELSON, G. M., FOLEY, T. A., HAMANN, B., AND LANE, D. Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics & Applications* 11, 3 (May 1991), 47–55.
- [41] NIELSON, G. M., AND FRANKE, R. Scattered data interpolation and applications: A tutorial and survey. In *Geometric Modeling: Methods and Their Applications*, H. Hagen and D. Roller, Eds. Springer, 1990, pp. 131–160.
- [42] O’ROURKE, J. Polyhedra of minimal area as 3D object models. In *Proc. of the International Joint Conference on Artificial Intelligence (1981)*, pp. 664–666.
- [43] POTTMANN, H. Interpolation on surfaces using minimum norm networks. *Computer Aided Geometric Design* 9 (1992), 51–67.
- [44] PREPARATA, F. P., AND SHAMOS, M. I. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.
- [45] RESCORLA, K.  $C^1$  trivariate polynomial interpolation. *Computer Aided Geometric Design* 4 (1987), 237–244.
- [46] SEDERBERG, T. W. Piecewise algebraic surface patches. *Computer Aided Geometric Design* 2 (1985), 53–59.
- [47] TURK, G., AND LEVOY, M. Zippered polygonal meshes from range images. In *Computer Graphics Proceedings (1994)*, Annual Conference Series. Proceedings of SIGGRAPH 94, pp. 311–318.
- [48] VELTKAMP, R. C. 3D computational morphology. *Computer Graphics Forum* 12, 3 (1993), 115–127.
- [49] WORSEY, A., AND FARIN, G. An n-dimensional Clough-Tocher interpolant. *Constructive Approximation* 3, 2 (1987), 99–110.