Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans[1] [2]

Chandrajit L. Bajaj[3] Fausto Bernardini[3] [4] Guoliang Xu[5]

Department of Computer Sciences
Purdue University

ABSTRACT

We present an efficient and uniform approach for the automatic reconstruction of surfaces of CAD (computer aided design) models and scalar fields defined on them, from an unorganized collection of scanned point data. A possible application is the rapid computer model reconstruction of an existing part or prototype from a three dimensional (3D) points scan of its surface. Color, texture or some scalar material property of the physical part, define natural scalar fields over the surface of the CAD model.

Our reconstruction algorithm does not impose any convexity or differentiability restrictions on the surface of the original physical part or the scalar field function, except that it assumes that there is a sufficient sampling of the input point data to unambiguously reconstruct the CAD model. Compared to earlier methods our algorithm has the advantages of simplicity, efficiency and uniformity (both CAD model and scalar field reconstruction). The simplicity and efficiency of our approach is based on several novel uses of appropriate sub-structures (alpha shapes) of a three-dimensional Delaunay Triangulation, its dual the three-dimensional Voronoi diagram, and dual uses of trivariate Bernstein-Bézier forms. The boundary of the CAD model is modeled using implicit cubic Bernstein-Bézier patches, while the scalar field is reconstructed with functional cubic Bernstein-Bézier patches.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; J.6 [Computer-Aided Engineering]: Computer-Aided Design.

Additional keywords: Geometric modeling, shape recovery, range data analysis, algebraic surfaces, triangulations, alpha-shapes.

[1] This work has been supported in part by NSF grant CCR 92-22467, AFOSR grant F49620-94-1-0080, ONR grant N00014-94-1-0370 and ARO grant DAAH04-95-1-0008.

[2] See also
`http://www.cs.purdue.edu/research/shastra/shastra.html`

[3] Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398 USA. Email: {`bajaj,fxb`}`@cs.purdue.edu`

[4] Additional partial support from CNR, Italy

[5] Computing Center, Academia Sinica, P. O. Box 2719, Beijing, 100080, P. R. China. Email: `xuguo@cs.purdue.edu`

INTRODUCTION

In this paper we present an approach for the reconstruction of a surface, and scalar fields defined over it, from scattered data points. The points are assumed sampled from the surface of a 3D object, and the sampling is assumed to be *dense* and *uniform* (these terms will be given a more precise meaning later in the paper).

Laser range scanners are able to produce a dense sampling, usually organized in a rectangular grid, of an object surface. Some models also allow to measure the RGB components of the color (i.e. three scalar fields) at each sampled point. When the object has a simple shape, this grid of points can be a sufficient representation. However, objects with a more complex geometry, e.g. objects with holes, handles, pockets, cannot be scanned in a single pass, and the various scans are not easy to merge [42]. Other applications, for example recovering the shape of a bone from contour data extracted from a CT scan, require reconstruction of a surface from data points organized in slices. The approach of considering the input points as *unorganized* has the advantage of generating cross-derivatives by a uniform treatment of all spatial directions.

The reconstruction problem we are considering may be formally stated as follows:

> Let an unorganized collection of points $P = \{(x_i, y_i, z_i)\} \subset \mathbf{R}^3$ and associated values $W = \{w_i\} \subset \mathbf{R}^1$, $i = 1 \ldots n$, be given. The points P are assumed sampled from a domain D in $\mathbf{R}^3$ (the boundary of a three-dimensional object) while the values W 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 (surface-on-surface) $S^F : f^F(x, y, z)$ on some domain that contains P such that, for $i = 1 \ldots n$:
>
> (a) $|f^D(x_i, y_i, z_i)| < \varepsilon^D$
>
> (b) $|f^F(x_i, y_i, z_i) - w_i| < \varepsilon^F$
>
> where ε^D and ε^F are user-defined approximation parameters. The user can also choose the degree of the Bernstein-Bézier polynomial patches used in the approximation.
>
> Additionally, generate different visualizations of the domain surface S^D and the surface-on-surface S^F.

In this paper we reconstruct the C^1 smooth domain S^D using a piecewise algebraic surface (the zero contour of a C^1 trivariate piecewise polynomial function). The surface is constituted by

barycentric implicit Bernstein-Bézier patches, which are guaranteed to be single-sheeted within each tetrahedron. We have also developed a method similar to the one described here, but based on tensor-product Bernstein-Bézier patches [5].

Some researchers have focused on reconstructing a piecewise-linear representation of the unknown surface D [38, 25, 13, 30, 43]. These papers provide a very nice survey of both the varied nature of applications and past approaches.

Related prior work [2, 6, 7, 15, 16, 27, 28, 35, 33] of fitting with smooth implicit surface patches, minimally all require an input surface triangulation of the data points. The surface fitting paper of [32] 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. The authors propose either a C^0 reconstruction algorithm based on an adaptive tetrahedral decomposition, or a scheme that uses tri-quadratic (degree six) tensor product implicit surface patches with a Powell-Sabin type split to achieve C^1 continuity.

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, in the same time it also computes a C^1 smooth approximation S^F of the sampled scalar function.

A different, three-step solution is described in papers [30, 31, 29]. In the first phase, a triangular mesh that approximates the data points is created. In a second phase, the mesh is optimized with respect to the number of triangles and the distance from the data points. A third step constructs a smooth surface from the mesh.

If the surface S^D is given, the problem of constructing the scalar function S^F is known as surface interpolation on a surface, and arises in several application areas, e.g. in modeling and visualizing the rain fall on the earth, the pressure on the wing of an airplane or the temperature on the surface of 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 [9]. A number of methods have been developed since then for its solution (for surveys see [10, 26, 37, 34]). Most of the proposed approaches interpolate scattered data over planar or spherical domain surfaces. In [12] and [35], the domain surface is generalized to a convex surface and a topological genus zero surface, respectively. Pottmann [39] presents a method which does not possess similar restrictions on the domain surface but requires it to be at least C^2 differentiable. In [11] the C^2 restriction is dropped, however the interpolation surface is constructed by transfinite interpolation using non-polynomials. A similar non-polynomial transfinite interpolant construction is used in [36], while the interpolation scheme in [41] requires at least C^4 continuity. Another approach, based on interpolation with cubic (for C^1) or quintic (for C^2) polynomials, is described in [8].

1 OVERVIEW OF THE ALGORITHM

Our algorithm consists of the following three phases:

1. **Preprocessing:** Preprocess the data points so that a *signed-distance* function is defined and efficiently computable. I.e., given a query point q, the function must return the approximate distance of the point from the domain surface S^D, with a positive sign if q lies outside the object, and a negative sign otherwise. We use α-shapes [21] to compute a piecewise linear approximation of the domain S^D, from which the approximated signed distance is computed. Details on this part are given in Section 2.

2. **Approximation:** Incrementally decompose the space into tetrahedra. For each tetrahedron τ that contains a portion of the domain D, compute Bernstein-Bézier trivariate implicit approximants f_τ^D and f_τ^F for both the domain D and the field F, based on data points and on the signed-distance function described above. Then compute the errors of the approximants for the given data points, and repeat the process, refining at each step the decomposition, until the errors meet the specified requirements. The use of a global signed-distance function in the computation of the coefficients of each patch guarantees C^0 continuity of the reconstructed surfaces. We use an incremental 3D Delaunay triangulation scheme together with a suitable point-insertion scheme to avoid badly-shaped tetrahedra in the spatial decomposition. This part of the algorithm is further detailed in Section 3.

3. **Smoothing:** Use a Clough-Tocher 12-way split to make the reconstructed surfaces C^1-smooth. See Section 4 for details.

Our domain surface S^D and surface-on-surface S^F reconstruction scheme does not impose any convexity or differentiability restrictions on the original domain surface D or function F, except that it assumes that there is a sufficient sampling of the input point data to unambiguously reconstruct the domain surface D. While it is difficult to precisely bound the required sampling density, we address this issue in Section 2.4 and characterize the required sampling density in terms of an α-shape (subgraph of a Delaunay triangulation of the points) which matches the topology of the original (unknown) sampled surface D. Compared to the above methods our algorithm thus has the following advantages:

1. It unifies the reconstruction of the domain surface D and the scalar function F defined on the domain surface;

2. It is adaptive and approximates large dense data sets with a relatively small number of C^1 smooth patches.

Outline of the paper: The rest of our paper is as follows. In Sections 2, 3 and 4 we present a detailed description of Phases 1, 2 and 3 of the reconstruction algorithm as outlined above. In Section 5 we illustrate all the phases of the algorithm with the aid of a simple 2D example. In Section 6 we show some examples of reconstructed surfaces and surface-on-surfaces, and discuss possible directions of future investigation.

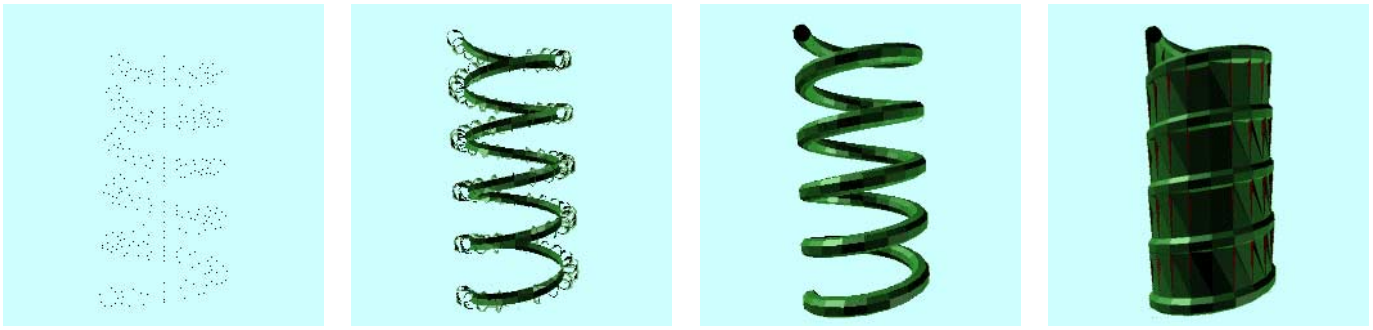
More details on the algorithm and additional examples can be found in [4].

2 PHASE 1: PREPROCESSING AND THE SIGNED-DISTANCE FUNCTION

As we mentioned in Section 1, our algorithm relies on the computation of the signed-distance $\delta(q)$ of a query point q from the domain surface D. The absolute value of $\delta(q)$ is defined as the Euclidean minimal distance of the point q from the domain surface D, while its sign is arbitrarily defined to be positive when q lies outside the object whose boundary is D, and negative otherwise.

In our implementation of the algorithm, we use α-shapes to compute a piecewise linear approximation L^D of the domain D, and make use of the associated data-structures (3D Delaunay triangulation and Voronoi diagram) to efficiently locate q w.r.t. the object and compute the associated signed-distance. An alternative method for computing an approximated signed-distance function, based on propagation of normals, is described in [30].

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.



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

Figure 1: A set of points in 3D (a), and three different α-shapes.

2.1 Delaunay Triangulations

Given a set P of points in $\mathbf{R}^3$ one can build a tetrahedralization of the convex hull of P, that is, a partition of $conv(P)$ into tetrahedra, in such a way that the circumscribing sphere of each tetrahedron τ does not contain any other point of P than the vertices of τ. Such a tetrahedralization is called a (3D) Delaunay triangulation and, under non degeneracy assumptions (no three points on a line, etc.) it is unique. Many different techniques have been proposed for the computation of Delaunay triangulations (see [19, 40, 18]). 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, approximating triangulation (see Section 3).

The algorithm we use is the randomized, incremental, flipping-based algorithm proposed in [22], with heed paid to robustness issues due to finite precision calculations [18]. 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 and the Delaunay property is re-established 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.

It is possible to build the Delaunay triangulation of a set of n points in $\mathbf{R}^d$ in $O(n \log n + n^{\lceil d/2 \rceil})$ 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 (for d = 2, 3) is much better than this theoretic bound (the actual running time depends on the distribution of points).

2.2 Voronoi Diagrams

Voronoi diagrams are well known tools in computational geometry (see [3] for a survey). They provide an efficient solution to the *Post Office Problem*, that is an answer to the query: what is the closest point $p \in P$ to a given point q? Voronoi diagrams are related to Delaunay triangulations by duality. It is easy to build a Voronoi diagram once one has the corresponding triangulation, and vice-versa. A Voronoi diagram is a partition of the space into convex cells. There is a cell for each point of $p \in P$, 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 one has to do to answer the 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 [14], one builds the point-location data-structure (called an *RPO-tree*, for Randomized Post Office tree) on top of the Voronoi diagram in $O(n^{2+\varepsilon})$ expected time, for any fixed $\varepsilon > 0$, and is then able to answer the closest-point query in $O(\log n)$ expected time. The data structure requires $O(n^{2+\varepsilon})$ space in the worst case. We use the *RPO-tree* data structure for our point location and signed-distance computations.

2.3 α-Shapes

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

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

(a) The size of σ is less than α and σ is conflict-free

(b) σ is a face of τ and $\tau \in \Sigma_\alpha$,

is called the α-shape of P.

α-Shapes have been introduced in the plane in [20] and then extended to the three-dimensional space in [21].

One can intuitively think of an α-shape as the subcomplex of $\mathcal{T}$ obtained in the following way: imagine that a ball-shaped eraser, whose radius is $\sqrt{\alpha}$, is moved in the space, assuming all possible positions such that no point of P lies in the eraser. The eraser removes all simplices it can pass through, but not those whose size is smaller than α. The remaining simplices (together with all their faces) form the α-shape for that value of the parameter α. Two extreme cases are the 0-shape, which reduces to the collection of points P, and the ∞-shape, that coincides with the convex-hull of P. Notice that there exists only a finite number of different α-shapes. The collection of all possible α-shapes of P is called the *family* of α-shapes of P (see Figure 1), and can be computed in time proportional to the number of simplices in $\mathcal{T}$. We use the α-shape computation for our generating an initial piecewise linear approximation L^D of the domain surface D (see Section 2.4).

2.4 Signed-Distance Computation

Obviously the domain surface D is unknown, so we need to build some suitable approximation of it to classify points as either internal or external to the object being reconstructed, and to compute a distance from it.

In the preprocessing phase the Delaunay triangulation of the set of input points P is computed, and then the Voronoi diagram and the family of α-shapes of P are constructed. During the process, the history DAG and the RPO-tree data structures are 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.

Tetrahedra in the Delaunay triangulation are classified as either internal or external (and assigned a corresponding sign) based on a particular α-shape chosen as a "good" linear approximation L^D to the surface to be reconstructed. The computation of the signed-distance is then reduced to locating the query point q in both the Delaunay triangulation, to decide its sign $s = \pm 1$, and in the Voronoi diagram, to find the closest point $p \in P$. The approximated signed-distance $s \cdot |pq|$ is then returned.

A difficulty in the process outlined above is the choice of a suitable value for α. We assume that the input data is dense enough so that there exists an α such that the α-shape approximates the object with the same topology as the original unknown surface D. In our current scheme a suitable α is selected interactively. The boundary of the selected α-shape must possess the following properties:

(a) It does not contain any singular (i.e. isolated) vertex;

(b) There are no "missing" edges, i.e. there can be missing triangles in the boundary, but if two adjacent triangles are missing, their common edge must be in the α-shape.

These properties make a slightly weaker condition than requiring that there exist an α-shape that correctly approximates the object *and* that has a *complete* boundary. In our experience it is sometime difficult to find an α value such that these stronger conditions are satisfied, even for "reasonably dense" samplings.

When an α-shape with the above properties is determined, it is easy to distinguish between internal and external tetrahedra in the underlying triangulation $\mathcal{T}$. One does a breadth first search on the dual graph of $\mathcal{T}$, starting with a tetrahedron that is known to be external (e.g. one that has a vertex at infinity) and continuing with adjacent tetrahedra. These tetrahedra are marked as external (positive sign) and put in a queue for further processing. When one hits a tetrahedron τ belonging to Σ_α, τ is marked as internal and not enqueued. The same happens when, visiting an adjacent tetrahedron τ of a positive tetrahedron σ, one finds that the common face (or all three edges of the common face) belongs to Σ_α. This means that going from σ to τ one crosses the boundary, so τ is marked as internal (negative sign) and not enqueued.

When the data points are not very dense or uniform, the error caused by using the approximated distance computation described above can be too large. In these cases, it is possible to improve the error by returning the distance of the query point from L^D, instead of P.

3 PHASE 2: INCREMENTAL REFINEMENT AND APPROXIMATION

In Phase 2 of the algorithm a 3D Delaunay triangulation $\mathcal{T}$ is initialized and incrementally refined, and C^0-continuous piecewise-polynomial functions (approximants) f^D and f^F are generated.

For each tetrahedron $\tau \in \mathcal{T}$ that contains a portion of D we compute two Bernstein-Bézier trivariate polynomials f^D_τ and f^F_τ, to approximate the part of domain surface and scalar field contained in τ, respectively. The coefficients of the polynomials are computed using data points within τ and the signed-distance function described in Section 2.4.

After computing the two polynomials, the errors of the approximants 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.

3.1 Bernstein-Bézier (BB) Form

Let $p_1,\ p_2,\ p_3,\ p_4 \in \mathbf{R}^3$ be affine independent. Then the tetrahedron τ with vertices $p_1,\ p_2,\ p_3$, and p_4, is $\tau = [p_1p_2p_3p_4]$. For any $p = \sum_{i=1}^{4} \alpha_i p_i \in \tau$, $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ is the barycentric coordinate of p. Let $p = (x, y, z)^T$, $p_i = (x_i, y_i, z_i)^T$. Then the barycentric coordinates relate to the Cartesian coordinates via the following relation

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

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

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

where

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

is a Bernstein polynomial, $|\lambda| = \sum_{i=1}^4 \lambda_i$ with $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$, $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ is the barycentric coordinate of p, $b_\lambda = b_{\lambda_1\lambda_2\lambda_3\lambda_4}$ (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 nonnegative integer components.

The points

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

are called the *regular points* of τ. The points $(p_\lambda, b_\lambda) \in \mathbf{R}^4$ are called *Bézier points* and their piecewise linear interpolation *Bézier net*.

The following lemma gives necessary and sufficient conditions for continuity.

Lemma 3.1 *([24]). Let $f(p) = \sum_{|\lambda|=n} a_\lambda B^n_\lambda(\alpha)$ and $g(p) = \sum_{|\lambda|=n} b_\lambda B^n_\lambda(\alpha)$ be two polynomials defined on two tetrahedra $[p_1p_2p_3p_4]$ and $[p'_1p_2p_3p_4]$, respectively. Then*

(i) f and g are C^0 continuous at the common face $[p_2p_3p_4]$ if and only if

$$a_\lambda = b_\lambda, \quad \textit{for all } \lambda = 0\lambda_2\lambda_3\lambda_4,\ |\lambda| = n \tag{2}$$

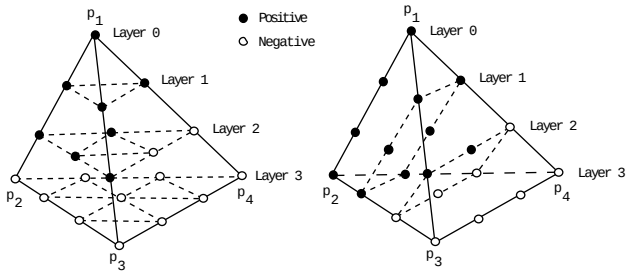


Figure 2: The layers of Bézier ordinates in a tetrahedron. (left) Three-sided patch. (right) Four-sided patch.

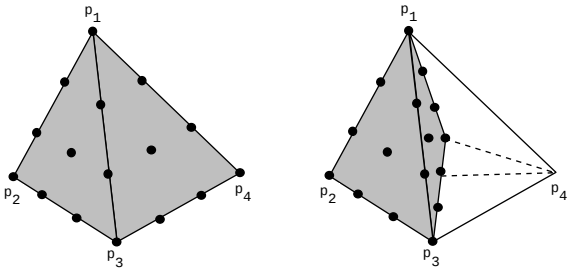


Figure 3: The splitting of a tetrahedron (right) into four sub-tetrahedra (left). Only one of the resulting sub-tetrahedra is shown.

(ii) f and g are C^1 continuous at the common face $[p_2p_3p_4]$ if and only if (2) holds and, for all $\lambda = 0\lambda_2\lambda_3\lambda_4$, $|\lambda| = n-1$,

$$b_{\lambda+e_1} = \beta_1 a_{\lambda+e_1} + \beta_2 a_{\lambda+e_2} + \beta_3 a_{\lambda+e_3} + \beta_4 a_{\lambda+e_4} \quad (3)$$

where $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)^T$ is defined by the following relation

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

The relation (3) will be called coplanar condition.

The following Lemmas give sufficient conditions for a patch to be *single-sheeted* (see [6] for proofs and further details).

Lemma 3.2 *Let $\tau = [p_1p_2p_3p_4]$. The regular points of τ 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 $[p_2p_3p_4]$ (see Figure 2). If the Bézier ordinates are all positive (negative) on layers $0, \ldots, k-1$ and all negative (positive) on layers $k+1, \ldots, n$ ($0 < k < n$), then the patch is single-sheeted (i.e., any line through p_1 and $p \in [p_2p_3p_4]$ intersects the patch only once).*

Lemma 3.3 *Let $\tau = [p_1p_2p_3p_4]$. The regular points of τ can be thought of as organized in quadrilateral layers, that we can number from 0 to n going from edge $[p_1p_2]$ to the opposite edge $[p_3p_4]$ (see Figure 2). If the Bézier ordinates are all positive (negative) on layers $0, \ldots, k-1$ and all negative (positive) on layers $k+1, \ldots, n$ ($0 < k < n$), then the patch is single-sheeted (i.e., any line through $p \in [p_1p_2]$ and $q \in [p_3p_4]$ 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 3.2 will be called *three-sided*; those satisfying the conditions of Lemma 3.3 will be called *four-sided*.

3.2 Outline of Phase 2 of the Algorithm

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 τ, such that $P \subset \tau$. Set $\mathcal{T} = \{\tau\}$ and V = vertices of τ. Mark τ as *new*.

2. For each *new* tetrahedron $\tau \in \mathcal{T}$, compute the signed-distance at all its regular points p_λ. If the values of $\delta(p_\lambda)$, $|\delta| = n$, do not satisfy either Lemma 3.2 or Lemma 3.3, then set $\vartheta_\tau^D = \vartheta_\tau^F = \infty$. Otherwise, compute local approximants

$$f_\tau^D(p) = \sum_{|\lambda|=n} b_\lambda^D B_\lambda^n(\alpha) \quad (4)$$

$$f_\tau^F(p) = \sum_{|\lambda|=n} b_\lambda^F B_\lambda^n(\alpha) \quad (5)$$

for the domain surface D and scalar field F as follows:

For the domain approximant, the coefficients b_λ^D are computed by first interpolating the computed values of the signed-distance function:

$$f_\tau^D(p_\lambda) = \delta(p_\lambda), \quad |\lambda| = n \quad (6)$$

The tetrahedron τ is then split into four sub-tetrahedra $\tau_1 \ldots \tau_4$ (see Figure 3) by joining the baricenter of τ with its four vertices (τ_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 τ. For these points we use the coefficients computed from (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 of the sub-tetrahedra are computed by solving the least squares problem

$$\begin{cases} f_{\tau_k}^D(p_i) = 0, & p_i \in P \cap \tau_k, k = 1 \ldots 4 \\ f_{\tau_k}^D(p_\lambda) = \delta(p_\lambda), & |\lambda| = n, \lambda_k \neq 0, k = 1 \ldots 4 \end{cases} \quad (7)$$

where we use the values of the signed-distance at regular points (of each sub-tetrahedron τ_k) in addition to the data points contained in τ. 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 τ:

$$f_\tau^D(p_i) = w_i, \quad p_i \in P \cap \tau \quad (8)$$

Notice that the field approximant is not globally continuous. Continuity will be achieved by averaging and interpolating values of the approximant at the vertices V of $\mathcal{T}$ in a subsequent phase, described in Section 4.

3. If the coefficients computed in the step above do not satisfy the conditions of either Lemma 3.2 or Lemma 3.3, set $\vartheta_\tau^D = \vartheta_\tau^F = \infty$. Otherwise, compute the approximation error for both functions:

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

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

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

$$\vartheta^D_{\sigma'} = \max_{\tau \in \mathcal{T}} \{ \vartheta^D_\tau \}$$

and

$$\vartheta^F_{\sigma''} = \max_{\tau \in \mathcal{T}} \{ \vartheta^F_\tau \}$$

4. If both $\vartheta^D_{\sigma'} < \varepsilon^D$ and $\vartheta^F_{\sigma''} < \varepsilon^F$ then the algorithm stops the incremental refinement phase and begins the smoothing phase. Otherwise, either σ' or σ'' is selected for further refinement (according to a user-definable strategy. E.g.: choose always σ' first, assigning priority to the surface, as variations of the scalar field F generally correspond to variations of the surface). The circumcenter q of the selected tetrahedron is computed and added to the set V 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 of σ is utilizing the empty sphere property of Delaunay triangulations and in general yields good aspect ratio tetrahedra in the final triangulation [17]). At the same time the subset $P \cap \tau$ of points that lie within each modified tetrahedron τ 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.

4 PHASE 3: ACHIEVING C^1 CONTINUITY VIA A 3D CLOUGH-TOCHER SCHEME

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 [44, 23]. In this scheme, one computes for each vertex in the original triangulation an average of the values of the functions f^D and f^F and their gradients, for all patches that share that vertex (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 twelve sub-tetrahedra 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 [44]), and joining these points with the original vertices. A cubic trivariate polynomial is built on each sub-tetrahedron. The coefficients of the twelve resulting patches are computed based on the value of the function at each vertex, the average gradient at vertices and mid-edge points, and the continuity constraint. The resulting patches are C^1 continuous and interpolate the averaged values and gradient of the functions.

Another trivariate Clough-Tocher scheme (see [1]) splits each tetrahedron into four sub-tetrahedra. However the interpolants in each sub-tetrahedra 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 interpolants, the higher order derivatives tend to be un-reliable 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 has been presented in [8], and its application to our method is described in [4].

5 A SIMPLE 2D EXAMPLE

We present in this section an example of the three phases of the algorithm. For presentation purposes, the steps are illustrated with the aid of a 2D example. The method is in fact perfectly suited for being applied in 2D reconstruction, and we chose to describe it only for the 3D case to keep the notation simple and because the most interesting applications arise from the study of fields on the surface of 3D objects. Restricting ourselves to a bi-dimensional example allows us to illustrate the various steps with pictures which we believe are more easily understood. The generalization of the techniques involved should be clear from the text.

In the following we refer to Figures 4(a)–(n). Figure (a) shows the sample points $P \in \mathbf{R}^2$. Figure (b) shows the associated function values W. The computed Delaunay triangulation and associated Voronoi diagram are depicted in Figure (c). These data structures will be used for fast point location in signed-distance computation. The chosen α-shape is shown in Figure (d). Four steps of the approximation phase are illustrated in Figures (e) though (i). Notice the adaptive subdivision of the plane. The implicit Bernstein-Bézier patches are shown in red. Empty triangles are light-blue and those containing a patch are grey. These triangles lie on the zero plane, so their intersection with the patches form the implicit curve $f^D = 0$. Figures (l) and (m) show the final reconstructed C^1 implicit patches, after Clough-Tocher subdivision, for both the domain and the scalar field. The zero contour of f^D is finally shown in Figure (n).

6 EXAMPLES AND CONCLUSIONS

Some examples of reconstruction of 3D objects and associated scalar fields are presented in this Section.

The data for the human femur in Figure 5, 9223 points, comes from contouring of a CT scan. The algorithm does not use the fact that the data is arranged in slices. The reconstructed C^1 surface is made by 400 cubic patches. The reconstruction algorithm took about 10 minutes on a SGI Indigo^2.

The engine in Figure 6 has been reconstructed from a data set containing 9800 points. The number of patches generated in the approximation phase is 382, with an error equal to 1/100 of the size of the object. Each patch is of degree 3, and is therefore defined by 20 coefficients. At the same time, an approximate C^1 scalar field (pressure form a simulated experiment) over the surface has also been computed. Several techniques can be used to visualize this surface-on-surface data. In Figure 6(c) we show iso-pressure regions. With the *normal projection* method, each point p on the domain surface S^D is projected in the direction normal to S^D, to a distance proportional to the value $f^F(p)$ of the field at that point. The projected surface is visible in transparency in Figure 6(d), with iso-contours of the pressure projected on it.

The data for the head of Spock is a subsampling (about 10^4 points have been used) of scan data obtained with a laser 3D digitizer. The reconstructed surface is constituted by 1100 cubic patches.

REFERENCES

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

[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. Power diagrams: properties, algorithms and applications. *SIAM J. Comput. 16* (1987), 78–96.

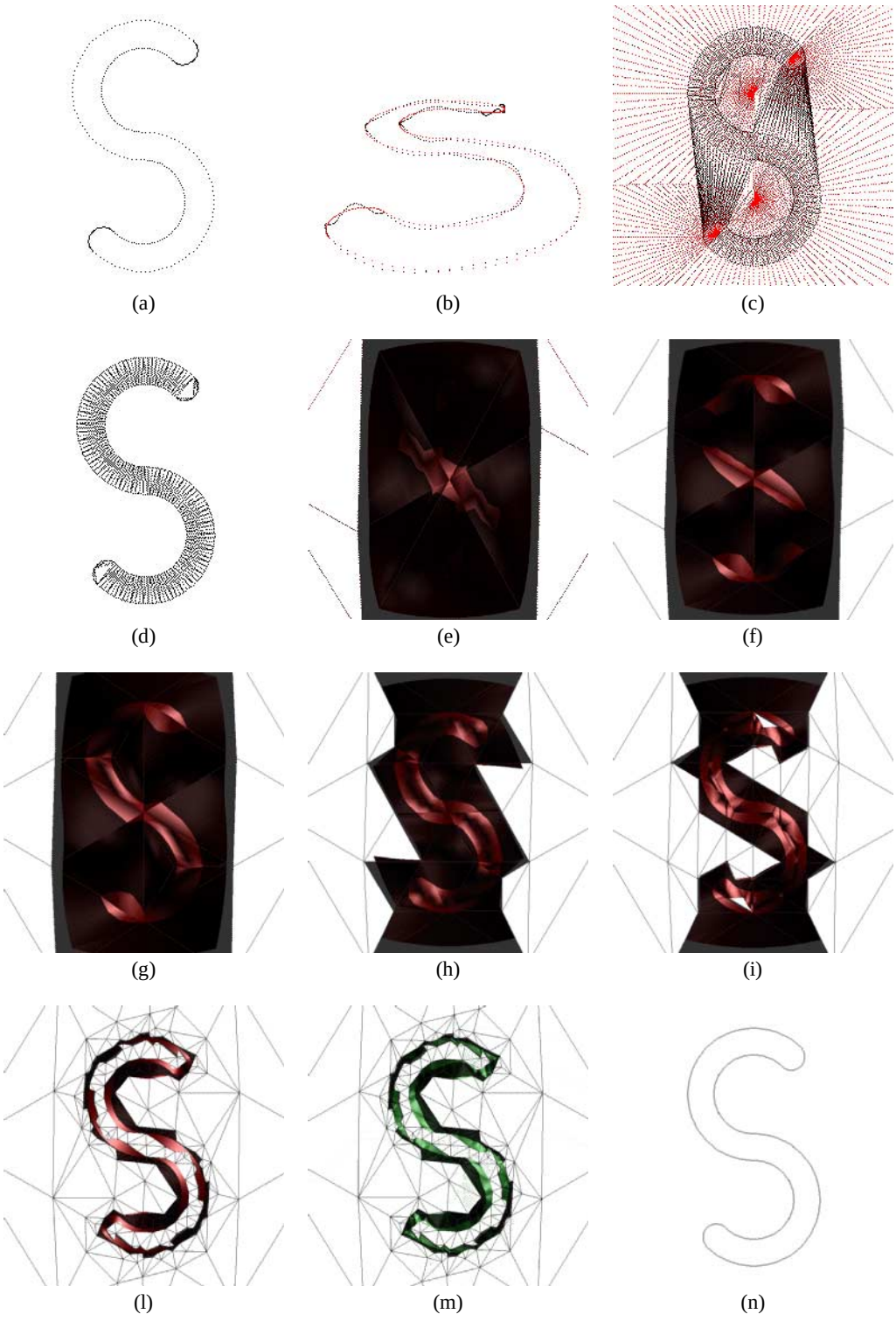
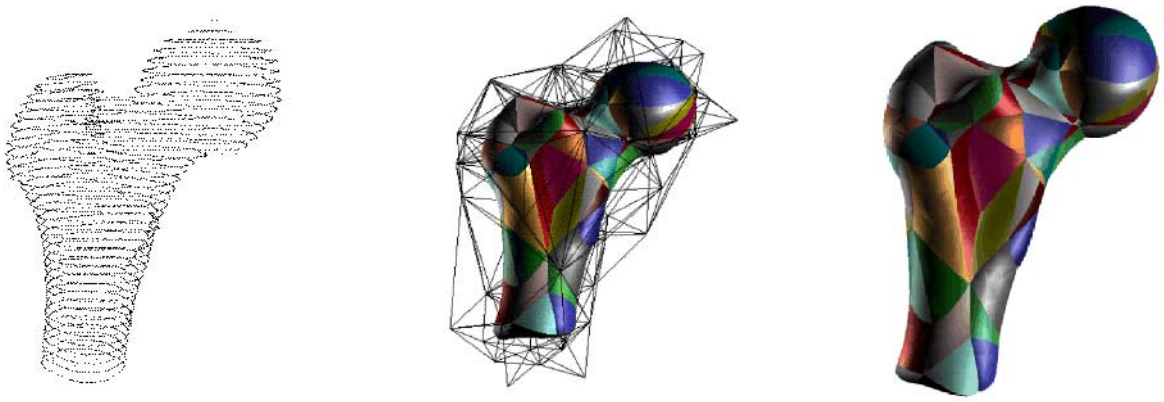
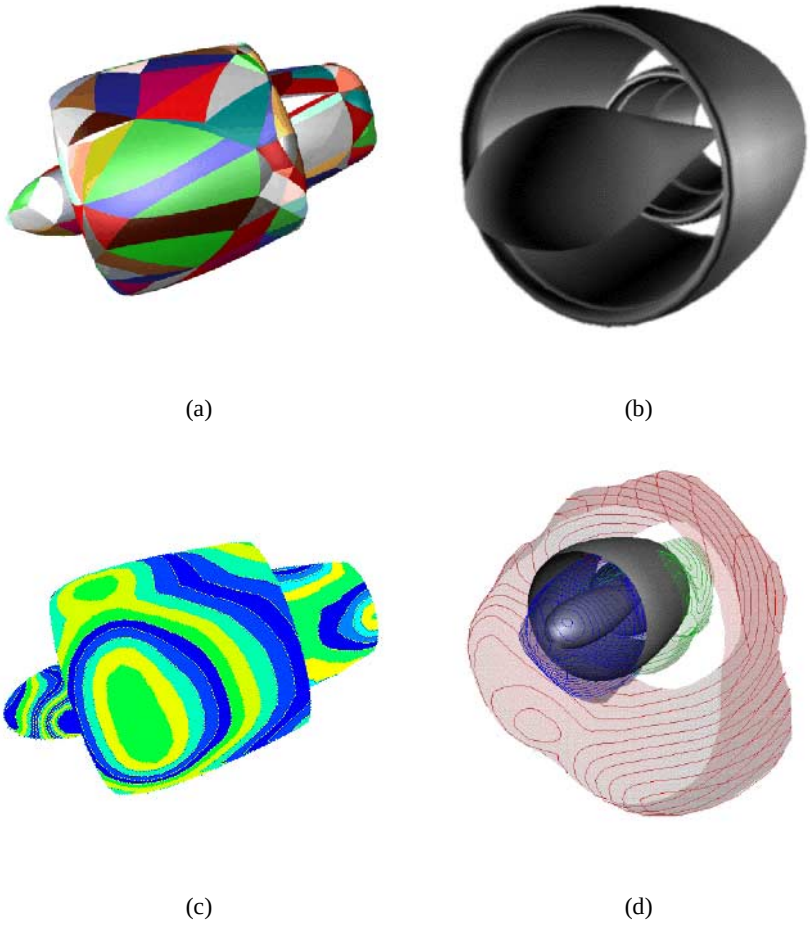


Figure 4: An example of reconstruction of the boundary of a two-dimensional shape and an associated scalar field.



(a) (b) (c)

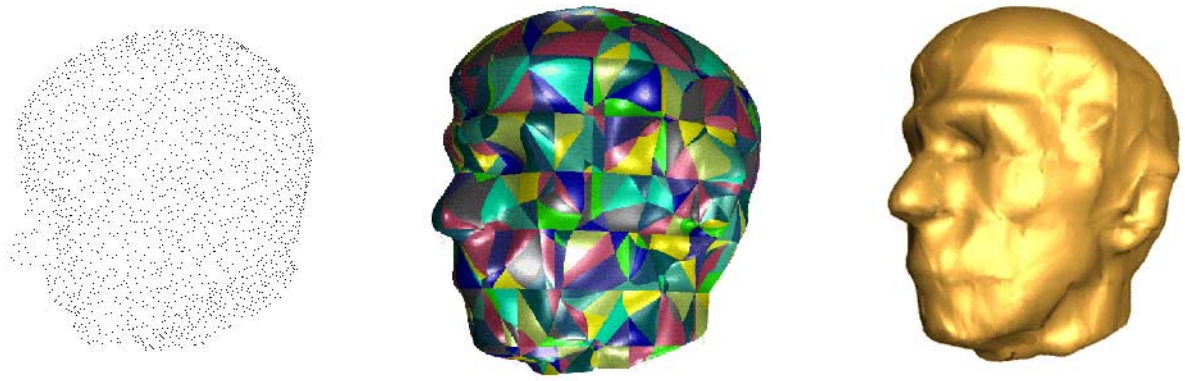
Figure 5: (a) Data set for the upper part of a human femur. Data from a CT scan. (b) Final decomposition (wireframe). (c) Reconstructed object.



(a) (b)

(c) (d)

Figure 6: A jet engine. (a) C^0 reconstructed domain. Patches are visible in different colors. (b) Reconstructed domain (after C^1 smoothing). (c) Iso-pressure contours and regions of a surface-on-surface pressure function displayed on the surface of the jet engine. (d) The reconstructed engine surface and visualization of the pressure surface function surrounding the jet engine using the normal projection method.



(a) (b) (c)

Figure 7: (a) A set of dense, scattered, noisy data points. (b) and (c) C^1 smooth reconstructed surface. In (b) patches have been randomly colored.

[4] BAJAJ, C., BERNARDINI, F., AND XU, G. Reconstruction of surfaces and surfaces-on-surfaces from unorganized weighted points. Computer Science Technical Report CSD-TR-94-001, Purdue University, 1994.

[5] BAJAJ, C., BERNARDINI, F., AND XU, G. Adaptive reconstruction of surfaces and scalar fields from dense scattered trivariate data. Computer Science Technical Report CSD-TR-95-028, Purdue University, 1995.

[6] BAJAJ, C., CHEN, J., AND XU, G. Modeling with cubic A-patches. *ACM Transactions on Graphics* (1995). To Appear.

[7] 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.

[8] 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. Beijing, China, 1994, pp. 19–29.

[9] BARNHILL, R. E. Surfaces in computer aided geometric design: A survey with new results. *Computer Aided Geometric Design 2* (1985), 1–17.

[10] 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.

[11] 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.

[12] 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.

[13] BOISSONAT, J. D. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics 3*, 4 (Oct. 1984), 266–286.

[14] CLARKSON, K. L. A randomized algorithm for closest-point queries. *SIAM J. Comput. 17* (1988), 830–847.

[15] 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.

[16] DAHMEN, W., AND THAMM-SCHAAR, T.-M. Cubicoids: modeling and visualization. *Computer Aided Geometric Design 10* (1993), 93–108.

[17] 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.

[18] DEY, T. K., SUGIHARA, K., AND BAJAJ, C. L. Delaunay triangulations in three dimensions with finite precision arithmetic. *Comput. Aided Geom. Design 9* (1992), 457–470.

[19] EDELSBRUNNER, H. *Algorithms in Combinatorial Geometry*, vol. 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[20] EDELSBRUNNER, H., KIRKPATRICK, D., AND SEIDEL, R. On the shape of a set of points in the plane. *IEEE Trans. on Information Theory 29*, 4 (1983), 551–559.

[21] EDELSBRUNNER, H., AND MUCKE, E. P. Three-dimensional alpha shapes. *ACM Transactions on Graphics 13*, 1 (Jan. 1994), 43–72.

[22] EDELSBRUNNER, H., AND SHAH, N. R. Incremental topological flipping works for regular triangulations. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.* (1992), pp. 43–52.

[23] FARIN, G. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design 3* (1986), 83–127.

[24] FARIN, G. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, 1990.

[25] FAUGERAS, O. D., HEBERT, M., MUSSI, P., AND BOISSONNAT, J. D. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics and Image Processing 25* (1984), 169–183.

[26] FRANKE, R. Recent advances in the approximation of surfaces from scattered data. In *Multivariate Approximation*, C.K.Chui, L.L.Schumarker, and F.I.Utreras, Eds. Academic Press, New York, 1987, pp. 275–335.

[27] GUO, B. Surface generation using implicit cubics. In *Scientific Visualizaton of Physical Phenomena*, N. M. Patrikalakis, Ed. Springer-Verlag, Tokyo, 1991, pp. 485–530.

[28] GUO, B. Non-splitting macro patches for implicit cubic spline surfaces. *Computer Graphics Forum 12*, 3 (1993), 434–445.

[29] 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, ACM SIGGRAPH, pp. 295–302.

[30] 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.

[31] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUELZLE, W. Mesh optimization. In *Computer Graphics Proceedings* (1993), Annual Conference Series. Proceedings of SIGGRAPH 93, ACM SIGGRAPH, pp. 19–26.

[32] 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.

[33] 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.

[34] NIELSON, G. M. Scattered data modeling. *IEEE Computer Graphics & Applications 13* (1993), 60–70.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] POTTMANN, H. Interpolation on surfaces using minimum norm networks. *Computer Aided Geometric Design 9* (1992), 51–67.

[40] PREPARATA, F. P., AND SHAMOS, M. I. *Computational Geometry: an Introduction*. Springer-Verlag, New York, NY, 1985.

[41] RESCORLA, K. C^1 trivariate polynomial interpolation. *Computer Aided Geometric Design 4* (1987), 237–244.

[42] TURK, G., AND LEVOY, M. Zippered polygonal meshes from range images. In *Computer Graphics Proceedings* (1994), Annual Conference Series. Proceedings of SIGGRAPH 94, ACM SIGGRAPH, pp. 311–318.

[43] VELTKAMP, R. C. 3D computational morphology. *Computer Graphics Forum 12*, 3 (1993), 115–127.

[44] WORSEY, A., AND FARIN, G. An n-dimensional clough-tocher interpolant. *Constructive Approximation 3*, 2 (1987), 99–110.