

Automatic Reconstruction of 3D CAD Models from Digital Scans ^{*}

Fausto Bernardini

Chandrajit L. Bajaj

Jindong Chen[†]

Daniel R. Schikore

Department of Computer Sciences
Purdue University[‡]

Abstract

We present an approach for the reconstruction and approximation of 3D CAD models from an unorganized collection of points. Applications include rapid reverse engineering of existing objects for use in a synthetic computer environment, including computer aided design and manufacturing. Our reconstruction approach is flexible enough to permit interpolation of both smooth surfaces and sharp features, while placing few restrictions on the geometry or topology of the object.

Our algorithm is based on alpha-shapes to compute an initial triangle mesh approximating the object's surface. A mesh reduction technique is applied to the dense triangle mesh to build a simplified approximation, while retaining important topological and geometric characteristics of the model. The reduced mesh is interpolated with piecewise algebraic surface patches which approximate the original points.

The process is fully automatic, and the reconstruction is guaranteed to be homeomorphic and error bounded with respect to the original model when certain sampling requirements are satisfied. The resulting model is suitable for typical CAD modeling and analysis applications.

^{*}Supported in part by AFOSR grant F49620-94-1-0080, NSF grant CCR 92-22467 and GER-9253915-02, and ONR grant N00014-94-1-0370

[†]Present address: Corporate Research & Technology, Xerox.
Email: jchen@cp10.es.xerox.com.

[‡]West Lafayette, Indiana
47907-1398 USA

{[bajaj](mailto:bajaj@cs.purdue.edu),[fxb](mailto:fxb@cs.purdue.edu),[jdc](mailto:jdc@cs.purdue.edu),[drs](mailto:drs@cs.purdue.edu)}@cs.purdue.edu
<http://www.cs.purdue.edu/research/shastra>

1 Introduction

The design, engineering and manufacturing planning of new products is more and more often carried out by computer simulation. A common need in this process is incorporating existing objects into this *electronic prototyping* environment, to reuse them as part of a new product, or to adapt and improve their design to meet new requirements.

The availability of fast and accurate geometric data acquisition devices, such as the *laser range scanner*, has made it relatively simple to acquire the spatial coordinates of a large set of points from the surface of a 3D object. Applications that would benefit from an efficient and reliable method for building a geometric model from this collection of measurements include:

Reverse engineering: Starting from an existing object, reconstruct a computer model of it, and then analyze and modify its design. Reverse engineering has relevant applications in the manufacture industry.

Shape analysis: Analyze the deformation of a mechanical part after a collision.

Authoring 3D virtual worlds:

Quickly build models of characters, actors, and spaceships from their real counterparts or from clay mock-ups.

3D fax: Scan an object, and transmit the digitized data on a phone line. The receiving station will reconstruct the model and manufacture a copy using a rapid prototyping technique such as stereo-lithography.

Tailor-fit modeling: Manufacture customized fashion apparels, helmets or prosthesis from a body scan.

Automatically reconstructing a CAD model of the object from a dense and uniform sampling of its surface is the subject of this paper. Among the qualities we would expect from such a model are the following:

- It matches the topological characteristics of the object;
- It is geometrically accurate;
- It can represent smooth, curvature continuous surfaces as well as sharp features such as corners and edges, common in manufactured parts;
- It is suitable to be used in successive phases of the design and simulation process.

We are interested in both the theoretical challenges that such a problem raises, and in practical methodologies which can be used in real-world applications. Previous research on this problem has mainly focused on reconstructing objects whose topology is known a priori. More recently, several methods have been proposed for the case of unknown topology. Many of these methods rely on heuristics to reconstruct spatial relationships between points. While these approaches have been shown to give good results on practical examples, there is no guarantee that they will not fail in producing a valid output. We give a formal characterization of sufficient conditions for the sampling.

Our study focuses on *dense*, unorganized data samplings. Last-generation devices are capable of measuring 10^4 to 10^5 points per second, with a resolution of 10^{-2} mm. 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. We will assume that the data comes in the form of an unorganized collection of x, y, z triples, and that no other geometric or topological information is available. This allows a unified treatment of different instances of the problem.

We are mainly interested in objects whose geometry is not easily specified as a combination of basic shapes. For example, many mechanical parts can be “disassembled” in a set of simple geometric entities (say parallelepipeds, cylinders, spheres etc.), combined via set operations (a representation called Constructive Solid Geometry, or CSG). For objects

like these, it makes sense to fit parts from this predefined small set so that their combination matches the sampled points. This problem involves *shape recognition* and *segmentation*. We will instead mainly deal with objects whose boundary is a “free form” surface, or a piecewise combination of smooth surfaces adjoining along sharp edges.

In summary, our contributions are the following:

1. We devise an efficient algorithm, based on alpha-shapes, capable of “connecting the dots”, by inferring spatial relationships between the sampled points. The algorithm automatically builds a triangle-mesh interpolating the data points. For an object whose “feature-size” is larger than some ρ , a ρ -dense sampling suffices to reconstruct a homeomorphic, distance-bounded model from the sampled points only.
2. We develop a fitting scheme for models with smooth faces and sharp features, based on a mesh reduction step followed by least-squares fitting of algebraic patches.

There are aspects of the reverse engineering problem that we have not investigated in this work. For example, data is subject to noise, especially in the vicinity of sharp features, and a reliable recovery of these features for segmentation purposes can be difficult using only “local” information. Also, when a complex, smooth surface is partitioned into patches, it would be helpful if the reconstruction algorithm could provide a “natural” partitioning, one for example that takes into account symmetries and other properties of the shape. A discussion of these and other related issues can be found in the recent review [53]. We provide further discussion of open problems and future work in Section 9.

2 3D Scanning Technologies

Typical 3D digitizers are based on touch probes, optical or range laser scanners, and acoustic or magnetic sensors.

Touch probes can be mounted on a 3D pantograph (see Figure 1a), and operated manually. Joint angles are measured by electronic or optical sensors, and these measures are combined and transformed into x, y, z coordinates. Automatic measurement machines are also widely used in the mechanical manufacturing industry, for example for quality control. In this configuration (Figure 1b), the mechanical probe is attached to a robot arm, and moved in contact with the object’s surface by specialized control software.

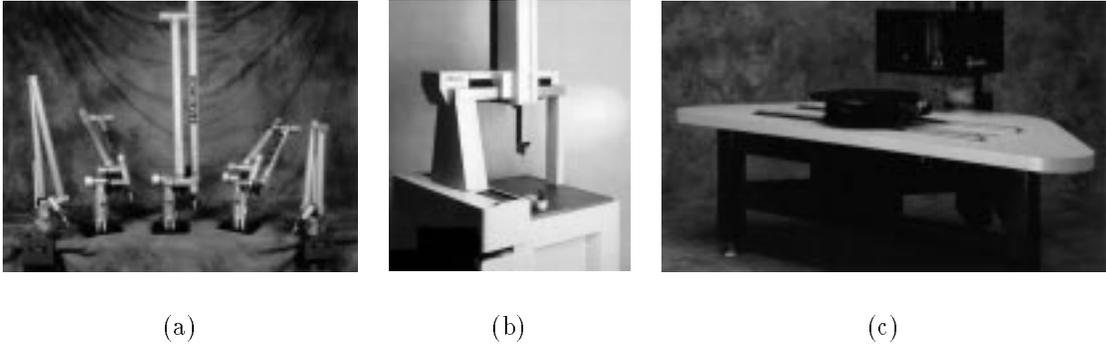


FIGURE 1: Several types of 3D digitizers. (a) Mechanical touch probes. (b) A computer-controlled Coordinate Measurement Machine. (c) A laser-range scanner, and the associated motion platform.

For some applications the object to be measured is mounted in a lathe stand, and the measuring probe scans its surface at points arranged in a cylindrical pattern.

Optical laser scanners shine a point or stripe of laser light on the object’s surface. One or more video sensors capture the point (or profile) from various viewpoints. The video image is digitized, and the position of each point is computed via triangulation. Objects are usually fixed to a platform whose position is controlled by precision servo motors. The accuracy of this motion is crucial in achieving low measurement errors. For a linearly translating platform one obtains a regular array of x, y, z measurements, or *range image*. Other typical platforms allow a rotating motion, yielding a cylindrical scan of the object. A picture of a commercial laser-stripe, two-sensor scanner is visible in Figure 1c. Typical spatial resolutions of such types of scanners range between $100\mu\text{m}$ and $500\mu\text{m}$, with a sampling speed of 15,000 points/sec.

The workspace of a fixed-head laser scanners is limited by the range of motion of the platform. Recent models are trying to overcome this limitation by mounting the laser scanning head on a precise pantograph. This allows to scan, with relative freedom of movements, objects of relatively large scale and with complex geometry, very much like one would spray-paint the object.

It is important to notice that, depending on the scanning technology used, the acquired data can have different characteristics. Mechanical, hand held probes are often used to get precise measurements of a relatively small number of points on important geometric features of the object. This set of data points are later connected together manually to form a wire-frame of curves, which can then be lofted. This pro-

cess is very time-consuming, and is not suitable for capturing the shape of free-form surfaces. Another mode of use is to simply “scribble” with the probe the object’s surface, and then use an automatic reconstruction algorithm to fit a surface to the sampled points.

With laser range scanners, when digitizing objects with a simple (say roughly cylindrical) geometry, a single scan might suffice to capture a regular grid of points on the whole object’s surface. These points can then be connected to form a triangle mesh. Since the triangle mesh can contain millions of triangles, it is usually simplified or approximated with parametric surface patches before being used as a model. Objects with a complex geometry, for example when holes and cavities are present, require more than one scan to allow the scanning head to “see” all of their surface. These different scans must then be co-registered, and converted into a single triangle-mesh. While methodologies for doing this have been proposed and shown to work well for some applications (see Section 3), they are not completely robust, in that they must deal with overlapping measurements that not always match completely, due to the limited accuracy of the scanning device.

Finally, it must be noted that additional geometric and topological information might be available. For example, one could record for each point the position of the scanning head, which can be used to compute a global, coherent orientation for the reconstructed surface. Often a measure of the confidence associated with each measurements can also be estimated, based on the angle of incidence that the sensor viewing direction forms with the surface normal (measurements are not very reliable when the camera sees the surface profile at a grazing angle). Also, many recent

laser scanners are coupled with an additional video-camera, which can capture surface color information.

3 Related Prior Work

Research on the 3D reconstruction problem has been done mainly in three areas: (a) numerical analysis and approximation; (b) geometric modeling and computer graphics; (c) computer vision.

When reconstructing a model representation from unorganized points, a major problem is that of inferring the topological genus of the object. Some of the approaches described in the following assume that the topological genus of the object is known *a priori*, and often restrict themselves to genus-zero objects.

Work in the field can be grouped into three broad categories: In *piecewise-linear reconstruction* the goal is constructing a polygonized (often triangulated) surface that interpolates or approximates the given points. Computational Geometry provides a rich framework of concepts and techniques to attack this problem.

Surface fitting is based on techniques from Computer Aided Geometric Design (CAGD) and Numerical Analysis. The reconstructed object is represented as a collection of surface patches (algebraic, NURBS, etc.).

A set of methods based on *physically-based modeling* have been developed and used, especially by the Computer Vision community. In these methods a surface is “deformed”, under the action of applied forces and internal reactions, until it approximates the data points.

In the rest of this chapter we give a short overview of earlier work in the field. The exposition is not meant to be exhaustive, but should give a clear idea of the range of techniques that have been used. Another review of the field can be found in [13].

3.1 Piecewise-Linear Reconstruction

O’Rourke [39] explored the use of polyhedra to represent the “most reasonable” reconstruction of an object from a set of points. In particular, he proposed polyhedra of minimal surface area as a “natural” model for a given set of points, and described an algorithm for the computation of an approximation of such polyhedra.

The problem of reconstructing a *polygon* of minimal perimeter having a given set of points as vertices is equivalent to the Euclidean Traveling Salesman problem, which is known to be NP-hard. The problem of computing a polyhedron of minimal surface area

is the 3D version of the minimal perimeter polygon, and is conjectured to be NP-hard as well.

O’Rourke proposed the following algorithm to compute an approximation of the minimal area polyhedron. The general strategy consists in starting with the convex hull of the given set of points, and then *shrinking* this surface to accommodate points that lie in its interior. At each step, he selects one interior point to be inserted in the current triangulated surface based on the distance to the closest face and other heuristics. Then he replaces the closest (triangular) face with a set of triangles that accommodate the new point and the boundary of the old face. Subsequently, he restores minimality of surface area in the neighborhood of the newly inserted vertex by flipping edges across adjacent faces. Contrary to TSP heuristics, this approximate algorithm is not guaranteed to yield a surface area within a fixed percentage of the minimal. This method is restricted to the reconstruction of objects of topological genus zero.

Boissonnat [12] proposed two methods to build a triangulation having the given points as vertices. The first method is “local” and surface-based, whereas the second one is “global” and volume-based.

Following his first approach, one starts with creating an edge between the two closest points. A third point is then chosen and added, so that a triangle is formed. Other points are successively added and new triangles are created, and joined to an edge of the current triangulation boundary, until all points have been included. For each edge E on the current boundary the new point p to be joined to E is selected among k closest neighbors according to a heuristic.

The second method is based on the idea of first computing a Delaunay triangulation of the convex hull of the set of points, and then *sculpturing* the volume by removing tetrahedra, until all points are on its boundary, or no tetrahedra can be further removed. Candidate tetrahedra are kept in a priority queue according to some geometric criterion, and removed only if the resulting polyhedron maintains manifold properties. It can be proved that any polyhedron of genus zero inside the Delaunay triangulation can be obtained by such a procedure [12]. However, depending on the sequence of tetrahedra removed by the algorithm, the sculpturing might get to a point in which no other tetrahedron can be removed, and yet some of the data points are still internal to the sculpted surface. Several geometric measures can be used to decide which tetrahedra are to be removed first. Boissonnat suggests using the maximum distance between the boundary faces of a tetrahedron and the associated parts of its circumscribing sphere. Tetrahedra with the largest value of this distance are

removed first.

Choi *et al.* [15], described a method to incrementally form a triangulation, starting from an initial triangle, based on the assumption that there exists a point from which all the points of the surface are visible. After a triangulation is built, it is improved by edge swapping based on a *smoothness* criterion.

Veltkamp [54] introduced a new geometric structure, called the γ -graph, which contains as a special case many well known geometric graphs, such as the Euclidean Minimum Spanning Tree, the Delaunay Triangulation, the Convex Hull and the Gabriel Graph. The γ -graph coincides initially with the convex hull, and is progressively *constricted* (i.e. tetrahedra having boundary faces are deleted) until the boundary of the γ -graph is a closed surface, passing through all the given points.

Hoppe *et al.* [32] compute a signed-distance function from the data points, and then use its zero-contour as an approximation of the object. First, for each data point $p_i \in P$, they compute a “tangent” plane, and the associated normal \hat{n}_i , based on best-fit of k neighbor points (the k -neighborhood of p_i). The problem is now assigning a consistent orientation to these planes. They build the *Riemannian Graph*, $RG(P)$ over P (two points $p_i, p_j \in P$ are connected by an edge in $RG(P)$ iff either p_i is in the k -neighborhood of p_j or p_j is in the k -neighborhood of p_i). Each edge (i, j) is assigned the weight $1 - |\hat{n}_i \cdot \hat{n}_j|$, and a minimum spanning tree is computed. Intuitively, this tree connects points that have close-to-parallel associated planes. They then orient the plane associated with the point with the largest z -value so that its normal points toward the positive z -direction, and propagate this orientation to other points traversing the minimum spanning tree. While there is no guarantee that the algorithm will find a coherent orientation, the traversing order implicit in the MST favors propagation across relatively smooth portions of the manifold, delaying more difficult areas of high curvature. Their paper shows, with several examples, that their heuristic yields good results in practice. Subsequently, the value of a signed-distance function δ is computed at all vertices of a grid of voxels as the distance of the vertex from the oriented plane associated with the closest point in P , with a sign depending on which side of the plane the point lies in. A *marching cubes* algorithm is then used to extract a piecewise-linear approximation of the zero contour of δ . In two subsequent steps, described in [33, 31], the constructed mesh is optimized (i.e., the number of triangles is reduced while the distance of the mesh from the data points is kept small) and then a piecewise smooth subdivision surface is

built on it (see also Section 3.2, page 6).

Turk and Levoy [52] proposed to “zipper” together several meshes obtained from separate 3D-scans of an object. In this way they can reconstruct a polygonal mesh approximating the surface of an object, even when a single scan does not suffice to capture its shape.

More recently, Curless and Levoy [17] presented an approach to merge several range images by scan-converting each image to a weighted signed-distance function, represented in a regular 3D grid. The range images are first co-registered, and a triangle mesh is built for each of them by connecting neighbor samples. Then for each image the contribution to a global signed-distance function is computed, by evaluating the distance of each voxel to the triangle-mesh along the direction of view of the sensor, and computing a weight based on the angle formed by the viewing direction and the surface normal (typically, range scanners are prone to larger measurement errors when the surface is viewed at a grazing angle). These values are incrementally added to previously accumulated results.

When all the range images have been integrated, a marching-cube algorithm is used to extract the zero-contour surface from the volume. The authors provide some details on how to fill holes (areas not scanned by the sensor) and on some implementation issues for time and space efficiency. This method is claimed to be “robust”, because the final surface is extracted from a globally defined function. The time required to merge 48 scans in a $407 \times 957 \times 407$ grid is reportedly above three hours on a 250MHz MIPS R4400 processor.

Alpha-shapes were introduced in the plane by Edelsbrunner *et al.* in [23] and then extended to higher dimensions [22, 24], as a geometric tool for reasoning about the “shape” of a set of points. An α -complex of a set of points P , for a given value of the parameter α , is a subset of the 3D Delaunay triangulation of P , and the corresponding α -shape is its underlying space. Intuitively, an α -shape can be obtained as follows: Consider a ball-shaped *eraser*, of radius α , and think of P as a set of points in space that the eraser cannot inter-penetrate. The 3D Delaunay triangulation of P is a simplicial complex formed by tetrahedra, triangles, edges and vertices (3-, 2-, 1- and 0-simplices, respectively). Imagine moving the eraser everywhere in space, removing all simplices that the eraser can pass through (remember that the eraser is constrained by the data points). All that is left after the erasing constitutes the α -shape of P .

Obviously, as α varies, one obtains different α -

shapes. For example, for $\alpha = 0$ the α -shape is P itself. For $\alpha = \infty$ one obtains the convex hull of P . Varying α from 0 to ∞ , a *finite* collection of α -shapes is obtained. Notice that in general an α -shape is a non-connected, non-regular (i.e., having solid as well as 2-, 1- and 0-dimensional parts) polytope, and therefore is not directly suitable for our purposes.

An extension of the α -shapes, called *weighted α -shapes* (see [22]), allows one to associate a weight to each data point. The weights can be used to capture different levels of detail in the sampling. Large weights can be assigned to points in areas of low sampling density, and small weights can be used in dense regions to compensate for a non uniform sampling and for different feature sizes. More details will be given in Section 4.

An approach based on alpha-shapes to define an approximate signed-distance function, followed by a piecewise-algebraic surface fitting, is described in [3] (see also Section 3.2, page 7).

3.2 Surface Fitting

We have grouped in this Section methods based on *approximating* the set of points with a piecewise polynomial, parametric or implicit surface.

An application of this method is described by Shmitt *et al.* [45]. The input points are assumed organized in a rectangular grid, and are adaptively fitted using Bernstein-Bézier parametric bi-cubic patches, joined to form a G^1 continuous surface. The approximation process begins with a rough approximating surface and uses subdivision to achieve the needed level of accuracy.

Moore and Warren [38] describe a method for fitting *algebraic* surfaces to scattered dense data. Their method is adaptive and able in principle to deal with complex geometry and topology.

The fitting begins with a uniform mesh of tetrahedral elements that fill a region containing the data points. Then for each element in the mesh that contains points, a surface patch that approximates the points is computed based on least squares fit of the data points and of *auxiliary* data. An element can be split into smaller elements if the approximation error is too large, and the process repeated for each sub-element. Finally, function values and the first k derivatives of each fitting surface are computed at each vertex of the final mesh, and averaged. A C^k interpolant is defined on each element, based on the averaged values (this process is called *free-form blending* by the authors).

It is known that least squares approximation of data points with algebraic patches might produce sur-

faces having extraneous parts. The auxiliary data mentioned above serves the purpose of avoiding extraneous surface sheets. This data is an approximate sampling of the *signed-distance* function $\delta(x, y, z)$.

Obviously, since the surface is unknown, the signed distance can only be estimated. They compute the absolute value of $\delta(p)$ as the distance of p from the closest data point, and assign a sign to it based on the following scheme. A tetrahedron is regularly subdivided into a mesh of smaller sub-tetrahedra. If the data points are dense enough, then sub-tetrahedra containing data points form a “layer” that divides the tetrahedron in two connected components. Vertices of the mesh in these two regions are assigned, arbitrarily, opposite signs. The auxiliary data is constituted by the approximate value of δ computed at vertices of this mesh of sub-tetrahedra. They prove that if the data in a tetrahedron can be approximated within a sufficiently small ε by a plane, then fitting with the auxiliary data produces a smooth, single-sheeted surface.

They give examples of C^0 reconstruction of surfaces and briefly discuss a C^1 method (non adaptive) based on biquadratic, tensor-product implicit patches.

The technique of Hoppe *et al.* [31] (see Section 3.1, page 5) starts with a triangulated surface mesh and produces a smooth surface based on the subdivision surface scheme of Loop [36]. Their method is based on minimizing an energy function that trades off conciseness and accuracy of fit to the data, and is capable of representing surfaces containing sharp features, such as creases and corners. The surface is represented as the limit of an infinite refinement process. While this approach appears promising for some applications, a non closed-form representation makes it difficult to apply standard techniques in later stages of analysis and design.

More recently, Eck and Hoppe [21] proposed an alternative surface fitting approach based on tensor-product B-spline patches. They start by using the signed-distance zero-surface extraction method of [32] (see also Section 3.1, page 5). An initial parameterization is built by projecting each data point onto the closest face. The method continues with building from the initial mesh a *base complex* (a quadrilateral-domain complex, with the same topology of the initial mesh) and a continuous parameterization from the base complex to the initial mesh, leveraging on the work of Eck *et al.* [20]. A G^1 network of tensor-product B-spline patches, having the base complex as parametric domain, is then fit to the data points, based on the scheme of Peters [41]. The fitting process is cast as an iterative minimization of a func-

tional, which is a weighted sum of the distance functional (the sum of square Euclidean distances of the data points from the surface) and a fairness functional (thin plate energy functional).

Bajaj *et al.* [3] used alpha-shapes to build an initial, piecewise-linear approximation of the shape. They then define a signed distance function based on the initial approximation, and fit C^1 -smooth implicit algebraic patches to the data points and samplings of the signed distance function. The fitting is done incrementally and adaptively, and can be extended to capture multiple scalar fields whose values are associated to the sampled points. Their method can be applied to objects of general genus.

Another interesting technique, based on region growing and restricted to functional surfaces, is described in [44].

3.3 Physically Based Modeling

Another class of algorithms is based on the idea of *deforming* an initial approximation of a shape, under the effect of external forces and internal reactions and constraints.

Terzopoulos *et al.* [50] used an elastically-deformable model with intrinsic forces that induce a preference for symmetric shapes, and apply them to the reconstruction of shapes from images. The algorithm is also capable of inferring non-rigid motion of an object from a sequence of images.

Pentland and Sclaroff [40] adopted an approach based on the finite element method and parametric surfaces. They start with a simple solid model (like a sphere or cylinder) and attach virtual “springs” between each data point and a point on the surface. The equilibrium condition of this dynamic system is the reconstructed shape. They showed how the set of parameters that describe the recovered shape can be used in object recognition.

Other physically based approaches are described in [43, 42, 34].

4 Preliminaries

4.1 Topological spaces, homeomorphisms, and manifolds

A *topological space* is a set S together with a collection \mathcal{U} of subsets of S (that is, \mathcal{U} is a subset of 2^S) satisfying the following conditions:

1. $\emptyset \in \mathcal{U}, S \in \mathcal{U}$.
2. If $U_1, \dots, U_n \in \mathcal{U}$ then $\bigcap_{i=1}^n U_i \in \mathcal{U}$.

3. Arbitrary unions of elements in \mathcal{U} lie in \mathcal{U} ; that is, if $\tilde{\mathcal{U}} \subset \mathcal{U}$, then $\bigcup_{U \in \tilde{\mathcal{U}}} U \in \mathcal{U}$.

The elements of \mathcal{U} are called *open sets* in S . The collection \mathcal{U} is called a *topology* on S . We often suppress the \mathcal{U} and simply refer to S as a topological space.

A map f from a topological space X to another topological space Y is *continuous* if every neighborhood of $f(p)$ in Y is mapped by f^{-1} to a neighborhood of p in X . If f is bijective, and if both f and f^{-1} are continuous, then f is a *homeomorphism*. Two topological spaces X and Y are *homeomorphic* if there exists a homeomorphism $f : X \rightarrow Y$.

In the following, we will restrict ourselves to subsets of the n -dimensional Euclidean space, $S \subset \mathbf{R}^n$. Let us define the following subspaces of \mathbf{R}^n , with origin o :

$$\begin{aligned} H^n &= \{x \in \mathbf{R}^n \mid x_n \geq 0\} \\ B^n &= \{x \in \mathbf{R}^n \mid \|x - o\| \leq 1\} \\ S^{n-1} &= \{x \in \mathbf{R}^n \mid \|x - o\| = 1\} \end{aligned}$$

Open, half-open, and closed n -balls are homeomorphic to \mathbf{R}^n , H^n and B^n , respectively. An $(n - 1)$ -sphere is homeomorphic to S^{n-1} .

A set in \mathbf{R}^n is *bounded* if it is contained in an open ball. An *open covering* of a topological space S is a collection $\mathcal{V} \subset \mathcal{U}$ such that $\bigcup_{V \in \mathcal{V}} V = S$. A space S is *compact* if every open covering has a finite sub-covering. A subspace of \mathbf{R}^n that is both closed and bounded is compact.

A k -manifold in \mathbf{R}^n ($n \geq k$) is a subspace that is locally homeomorphic to \mathbf{R}^k . A k -manifold with boundary is a subspace that is locally homeomorphic to either \mathbf{R}^k or the half-open k -ball H^k . Points with a neighborhood homeomorphic to H^k form the *boundary* of the manifold X , denoted $\text{bd}(X)$. The boundary of a k -manifold with boundary is a $(k - 1)$ -manifold without boundary.

4.2 Simplicial complexes

A k -simplex $\sigma_T = \text{conv}(T)$ is the convex combination of an affinely independent point set $T \subset \mathbf{R}^n$, $|T| = k + 1$. k is the *dimension* of simplex σ_T . A (geometric) *simplicial complex* K is a finite collection of simplices with the following two properties:

1. if $\sigma_T \in K$ then $\sigma_U \in K, \forall U \subset T$
2. if $\sigma_U, \sigma_V \in K$, then $\sigma_{U \cap V} = \sigma_U \cap \sigma_V$ (1 and 2 imply that $\sigma_{U \cap V} \in K$).

The underlying space of K is $[K] = \bigcup_{\sigma \in K} \sigma$. A subcomplex of K is a simplicial complex $L \subset K$.

4.3 Alpha-shapes

Alpha-shapes [23, 24] associate a mathematically defined meaning to the vague concept of *shape* of an unorganized set of points. Weighted alpha-shapes [22] are a generalization of alpha-shapes to sets of weighted points. In the following, we will shortly review definitions and properties of alpha-shapes. The presentation is adapted from [22]. Notice that although the exposition is for *unweighted* alpha-shapes, we will use the notation used in the more general weighted case. A weighted alpha-shape coincides with an unweighted alpha-shape when all weights are equal to zero. We restrict our presentation to the three-dimensional case. n -dimensional weighted alpha-shapes are described in the cited reference [22].

In the following we will sometimes regard a sphere of radius ρ centered in p as a weighted point p of weight $w_p = \rho^2$. We define the power distance of a point x from a weighted point p as

$$\pi_p(x) = \|p - x\|^2 - w_p$$

where $\|p - x\|$ is the Euclidean distance between p and x . A geometric interpretation of the power distance is the following: If weighted point p represents a sphere of center p and radius $\sqrt{w_p}$, then $\pi_p(x)$ is the square of the length of a tangent line segment from x to the sphere (see Figure 2).

Let $P \subset \mathbf{R}^3$ be a finite set of points (general position is assumed implicitly throughout the paper), $|P| \geq 4$, and \mathcal{T} its Delaunay triangulation. For every simplex $\sigma_T \in \mathcal{T}$, let y_T be the smallest sphere (weighted point) such that $\pi_{y_T}(p) = 0, \forall p \in T$. If $|T| = 4$ there is only one such sphere y_T , the circumsphere of σ_T . If $|T| = k + 1 < 4$ there are infinitely many such spheres, but only one has minimum ra-

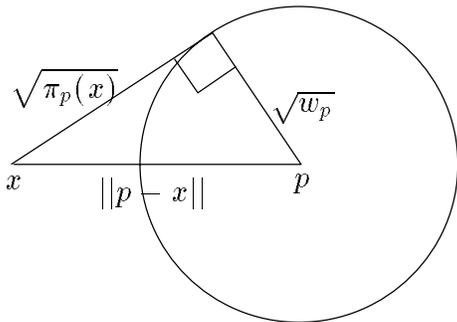


FIGURE 2: Power distance of a point x from the weighted point p .

dius. The center of y_T is located at the intersection of the *chordale* of T (see Figure 3)

$$\chi_T = \bigcap_{p,q \in T} \chi_{p,q}, \quad \chi_{p,q} = \{x \in \mathbf{R}^3 \mid \|p - x\| = \|q - x\|\}$$

with the orthogonal k -flat $\text{aff}(T)$. Let ρ_T be the radius of y_T , and call $w_{y_T} = \rho_T^2$ the *size* of the k -simplex σ_T . Notice that the size of a 0-simplex is 0. The size of simplices satisfies the following monotonicity property: if $U \subset T$ then $w_{y_U} < w_{y_T}$, that is the size of a proper face of a simplex is smaller than the size of the simplex itself.

A point $q \in P - T$ is a *conflict* for y_T if $\pi_{y_T}(q) < 0$, and y_T is *conflict-free* if it has no conflicts. Obviously, all 3-simplices $\sigma_T \in \mathcal{T}$ are conflict-free, but a k -simplex, $k < 3$, can have conflicts.

Definition 4.1 *The alpha-complex of P is the sub-complex Σ_α of \mathcal{T} formed by all simplices σ_T such that:*

- (a) *The size of y_T is less than α and y_T is conflict-free, or*
- (b) *σ_T is a face of σ_U and $\sigma_U \in \Sigma_\alpha$.*

The underlying space \mathcal{W}_α of Σ_α , called alpha-shape, is a polytope, which can be non-connected and different from the closure of its interior (i.e. it may contain parts of heterogeneous dimensionality).

It can be proved (see [22]) that the following is an alternative definition of alpha-shapes:

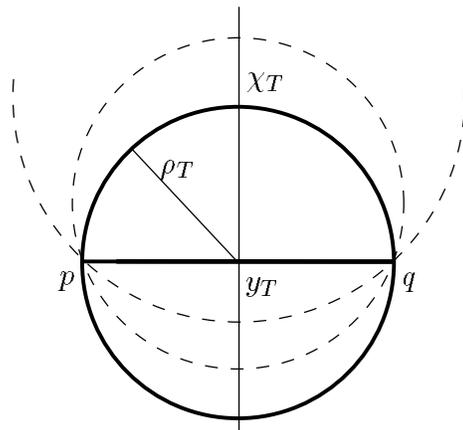


FIGURE 3: The collection of spheres containing the two vertices of the 1-simplex $T = \{p, q\}$. The sphere y_T of minimum radius ρ_T is drawn in bold.

Definition 4.2 Consider a subset $T \subseteq P$, with $|T| = k + 1 \leq 3$, and the k -simplex σ_T . Let us call σ_T α -exposed if there exists a weighted point x , of weight $w_x = \alpha$ (that is, a sphere of radius $\sqrt{\alpha}$), such that

$$\pi_x(p) = \begin{cases} = 0 & \forall p \in T \\ > 0 & \forall p \in P - T \end{cases}$$

The alpha-shape \mathcal{W}_α of P is a polytope whose boundary is the union of all α -exposed simplices spanned by subsets $T \subseteq P$, $|T| \leq 3$. The interior of \mathcal{W}_α is formed by those components of \mathbf{R}^3 bounded by collections of α -exposed 2-simplices σ_T , such that σ_T is α -exposed only on one side (i.e. there exists only one weighted point of weight α that exposes σ_T). The interior points of \mathcal{W}_α lie on the side of σ_T that is not α -exposed.

4.4 Bernstein-Bézier Forms

Any polynomial of degree n can be expressed as a Bernstein-Bézier form (BB-form) over a tetrahedron τ . An algebraic patch is the zero-set of a polynomial, restricted to the supporting tetrahedron. The Bernstein-Bézier form is particularly suitable to the representation of piecewise algebraic surfaces as it allows to express derivative continuity between patches with simple, geometrically intuitive constraints. The shape of each patch can be locally controlled by adjusting a net of *control points*. In this section we review definitions and basic properties of BB-forms, and introduce A-patches, algebraic patches that are guaranteed to be single-sheeted and singularity-free.

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, 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:

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

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

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

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 , $b_\lambda = b_{\lambda_1 \lambda_2 \lambda_3 \lambda_4}$ (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 τ . 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 4.1 (see [25]) 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

$$a_\lambda = b_\lambda, \quad \text{for all } \lambda = 0\lambda_2\lambda_3\lambda_4, \quad |\lambda| = n \quad (2)$$

- (ii) f and g are C^1 continuous at the common face $[p_2 p_3 p_4]$ if and only if (2) holds and, for all $\lambda = 0\lambda_2\lambda_3\lambda_4$, $|\lambda| = n - 1$,

$$b_{\lambda+\epsilon_1} = \beta_1 a_{\lambda+\epsilon_1} + \beta_2 a_{\lambda+\epsilon_2} + \beta_3 a_{\lambda+\epsilon_3} + \beta_4 a_{\lambda+\epsilon_4} \quad (3)$$

where $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)^T$ are the barycentric coordinates of p' w.r.t. $[p_1 p_2 p_3 p_4]$, defined by the following relation

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

The relation (3) will be called coplanar condition.

4.5 Definition and Properties of A-Patches

The success of parametric surfaces in geometric modeling is due to the ease of evaluation and local control they offer. Non-Uniform Rational B-Spline (NURBS) surfaces have become a fairly standard representation of free-form surfaces in commercial CAD packages. However, parametric surfaces have some important limitations, such as the fact that intersection and offset operations can produce results that are not representable exactly in parametric form [2].

In recent years, there has been an increasing attention to alternative forms of surface modeling. In particular, piecewise algebraic *implicit* surface representations have many appealing properties.

An algebraic surface [49, 47, 48] 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 [2]. By using the Bernstein-Bézier form to represent each polynomial, one inherits the large wealth of useful properties that have made this representation so popular in the parametric surface domain.

The main shortcoming of algebraic patches is that, in general, the zero-set of a polynomial can have more than one real sheet, and may contain singular points. Researchers have therefore looked at conditions on the weights that guarantee that the patch is single-sheeted and singularity-free (or *smooth*).

Sederberg [49] showed that, if the coefficients of the BB-form on the lines parallel to one edge of the tetrahedron all increase (or decrease) monotonically in the same direction, then any line parallel to that edge will intersect the surface patch at most once. Guo [26] treats the same problem by enforcing monotonicity conditions on a cubic polynomial along the direction from one vertex to a point on the opposite face of the vertex. From this he derives the condition $a_{\lambda - e_1 + e_4} - a_\lambda \geq 0$ for all $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$ with $\lambda_1 \geq 1$, where a_λ are the coefficients of the cubic BB-form and e_i is the i -th unit vector. This condition is difficult to satisfy in general, and does not in general avoid singularities on the zero-contour.

Another problem is how to “stitch” together a collection of patches so that they join with the desired derivative continuity.

For this problem, partial solutions have been given by Dahmen [18] using quadric patches, Dahmen and Thamm-Schaar [19], Lodha [35], and Guo [26, 27] using cubic patches and Bajaj and Ihm [6] using quintic for convex triangulations and degree seven patches for arbitrary surface triangulations. All these papers provide heuristics to overcome the multiple-sheeted and singularity problems of implicit patches.

All these methods use variations of the scheme described in [18] of building a surrounding *simplicial hull* of a given triangulation. Such a construction is

nontrivial and none of the papers cited enumerate the exceptional cases (possible even for convex triangulations) nor provide solutions to overcome them.

Related papers on approximating scattered data using implicit algebraic patches include [1, 35, 38].

More recently, Bajaj, Chen and Xu [4, 5, 14] have described a new class of algebraic patches, called *A-patches*. A-patches are guaranteed, by imposing constraints on the values of their weights, to be non-singular (except where needed, see below) and single-sheeted. A simplicial hull construction algorithm for C^1 surfaces, using cubic patches, or C^2 surfaces, using quintic patches, is also provided. Exceptional cases are individuated, and heuristic solutions to overcome them provided. Here we shortly review definitions and properties of A-patches. A more detailed discussion can be found in the cited references.

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 (see Figure 5).

Lemma 4.2 *Let $\tau = [p_1 p_2 p_3 p_4]$. 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 $[p_2 p_3 p_4]$ (see Figure 4). 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 4.3 *Let $\tau = [p_1 p_2 p_3 p_4]$. 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 $[p_1 p_2]$ to the opposite edge $[p_3 p_4]$ (see Figure 4). 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 4.2 will be called *three-sided*; those satisfying the conditions of Lemma 4.3 will be called *four-sided* (Figure 5). See [5] for proofs and further details.

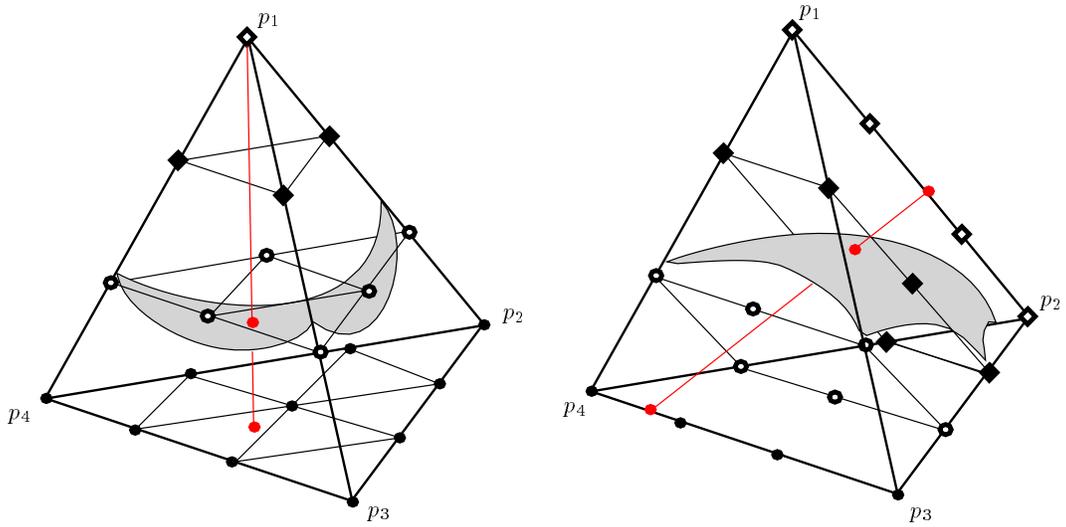


FIGURE 4: The layers of Bézier ordinates in a tetrahedron. (left) Three-sided patch. (right) Four-sided patch.

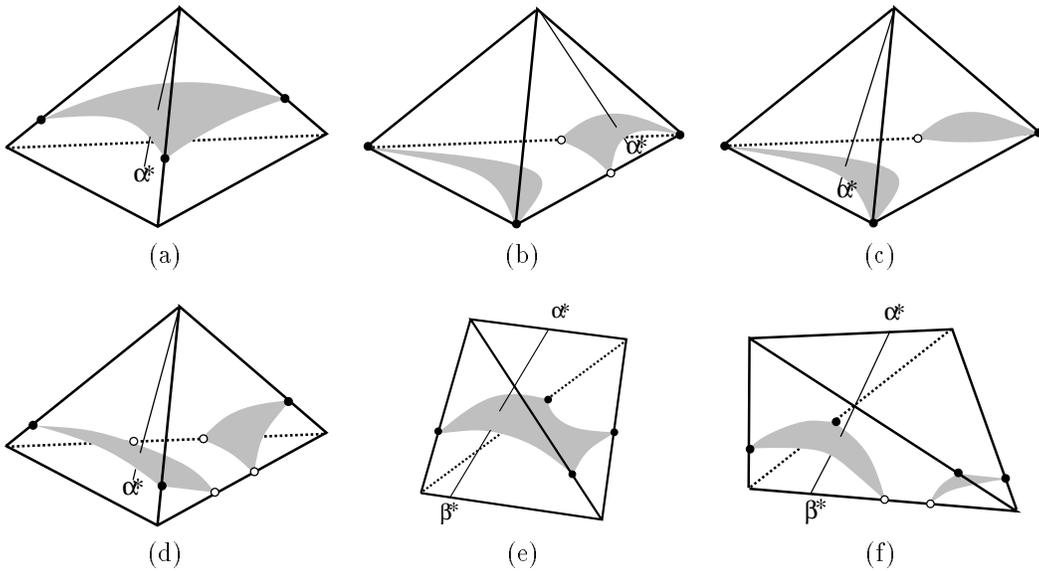


FIGURE 5: Different types of patches. (a), (b), (c) and (d) Three-sided patches. (e) and (f) Four-sided patches.

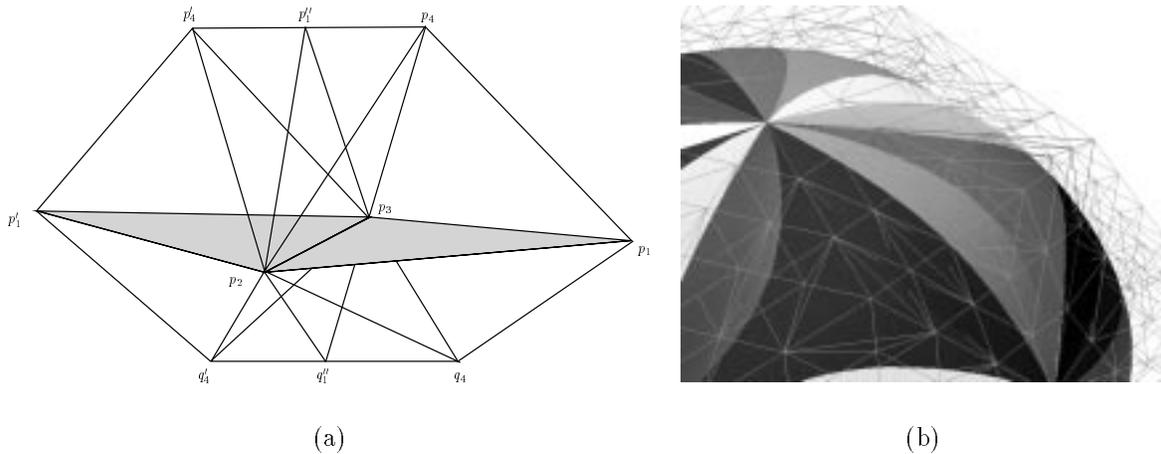


FIGURE 6: (a) Construction of a simplicial hull. The triangles $[p_1p_2p_3]$ and $[p'_1p_2p_3]$ belong to the initial triangle mesh. Vertices p_4, p'_4, q_4, q'_4 have been introduced to form the four *face-tetrahedra* (on the two opposite sides of each original triangle). Vertices p''_1 and q''_1 are needed to form the four *edge-tetrahedra*. (b) An example of simplicial hull. Note that for each tetrahedron the net of control points is also shown. The net of one of the face tetrahedra is highlighted in red. Patches have been randomly colored for clarity.

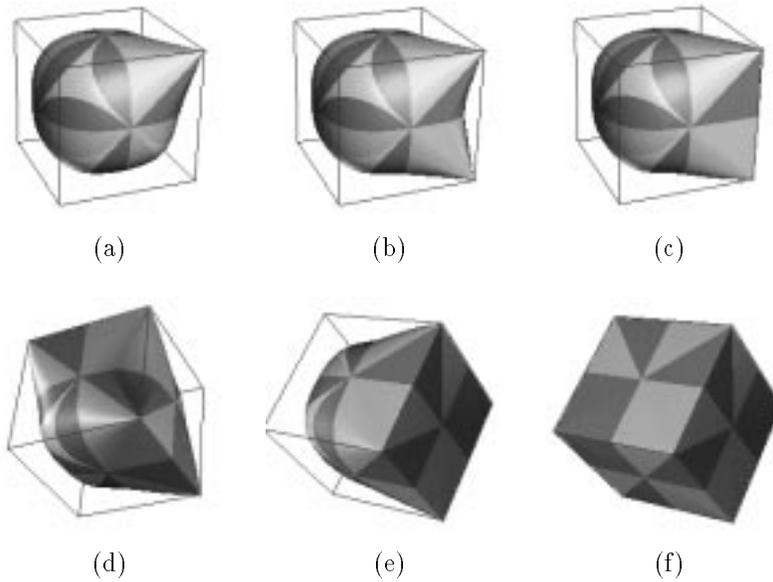


FIGURE 7: Modeling with singular A-patches. (a) Interpolating a vertex with a singular point. (b) Interpolating two vertices. (c) Interpolating an edge with a singular edge on the surface. (d) Interpolating two edges. (e) Interpolating a face of a cube. (f) The A-patch surface degenerates into the cube. All the edges are now singular.

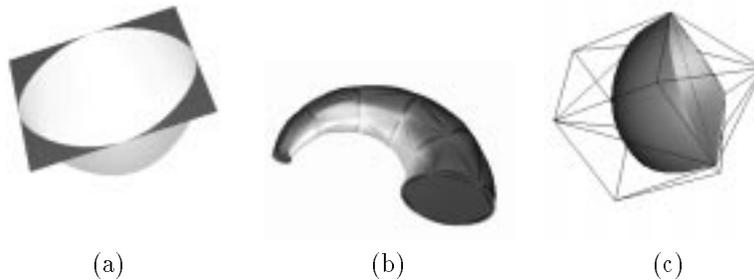


FIGURE 8: Examples of simple objects with sharp features modeled with singular A-patches.

4.6 Polyhedra “Smoothing” with A-Patches

An A-patch is defined inside a supporting tetrahedron. When modeling a piecewise algebraic surface, we need to “stitch” together a collection of patches, with the desired continuity properties. This raises two problems: (a) How to build a supporting tetrahedral mesh, i.e. a simplicial complex such that A-patches can be defined inside each 3-simplex to form a continuous (open or closed) surface. (b) How to set the weights of each patch so that the surface has the desired derivative continuity properties.

One possibility is to build a 3D triangulation of a convex polyhedron enclosing the surface to be modeled. One can for example define a finite set of points P and compute the 3D Delaunay triangulation \mathcal{T} of P . The subset of tetrahedra $\tau \in \mathcal{T}$ that intersect the surface will be used as support mesh. With this construction, no vertex of the support mesh will in general lie on the surface. The weights of each patch can be set so that the patch approximates point data within the tetrahedron, for example by solving a least-squares problem, and constrained to satisfy the single-sheeted conditions stated above. We used this type of construction to build an adaptive approximation of a trivariate signed-distance function in [3].

Another approach starts with a triangulated two-manifold that approximates the surface. One then builds two types of tetrahedra: (a) For each triangle of the mesh, two *face*-tetrahedra are created, one on each side of the triangle, and (b) For each edge of the mesh, four *edge*-tetrahedra are introduced, bridging the gaps between the four face-tetrahedra which share that edge (see Figure 6(a)). The tetrahedral mesh obtained with this process is called *simplicial hull* (see Figure 6(b) for an example), and details on its construction can be found in [5, 14]. This method is used in Section 8 to selectively smooth and approx-

imate a dense mesh of triangles.

Once the simplicial hull is constructed, we need to set the weights of each patch so that the surface is C^1 (or locally C^0 in the presence of a sharp feature) and the collection of patches:

1. Interpolate the vertices (and optionally the associated normals) of the triangle mesh;
2. Approximate other data points.

C^0 and C^1 features can be mixed into the same model, by appropriately setting weights and allowing patches with singular vertices/edges [14]. Figures 7 shows how C^0 and C^1 features can be mixed to interpolate some of the vertices, edges and faces of a cube. Other examples of sharp features modeled with singular A-patches are illustrated in Figure 8.

5 Overview of the Reconstruction Algorithm

An example of the reconstruction process is shown in Figure 9. The simple object shown will be used as a running example in the following Sections. Our algorithm is based on the following three phases:

1. Build an initial triangle mesh that interpolates all data points, approximating the object shape (Figure 9 (a)-(c)). Our approach (described in Section 6) is based on alpha-shapes [24], and is capable of automatically selecting an optimal alpha value and improving the resulting mesh in areas of insufficient sampling. We also present a theorem stating sufficient conditions on the sampling that guarantee a homeomorphic, error bounded reconstruction.

The resulting triangle mesh can be used to estimate normals at smooth vertices (by averaging

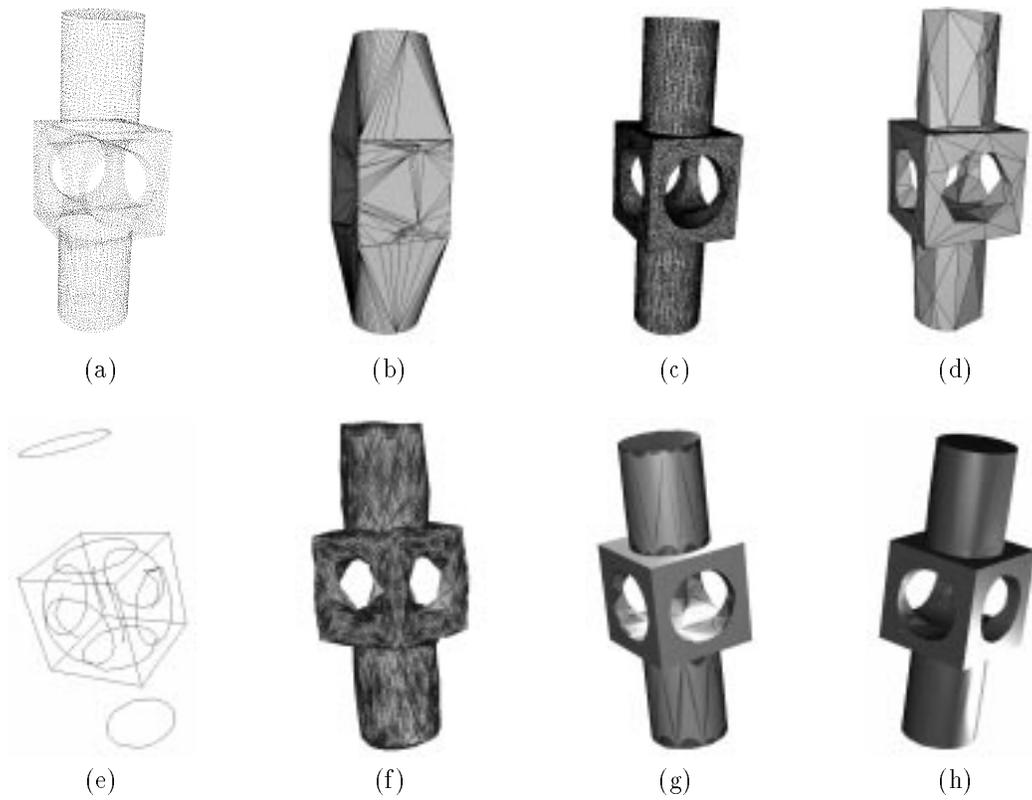


FIGURE 9: The complete reconstruction process. (a) Point sampling. (b) 3D Delaunay triangulation. (c) Alpha-solid. (d) Simplified mesh. (e) Sharp features. (f) Support mesh. (g) A-patches. (h) Reconstructed model.

the normals of incident triangles) and to detect sharp features (by looking at the dihedral angle formed by two adjacent triangles). Notice that for the dense surface sampling that we are interested in, these estimates are usually quite accurate. The use of more complex and accurate sharp-feature detection strategies will be investigated in the future.

2. Simplify the mesh to reduce the number of triangles, while guaranteeing good aspect-ratio of triangles, bounded distance of the data points from the reduced mesh, and feature preservation (Figure 9 (d)). The technique used in our paper has been extended from [8]. The edges and vertices of the reduced mesh are “tagged” as either *smooth* (the surface is C^1 continuous across it) or *sharp* (only C^0 continuity), and vertices are classified according to the type of incident edges and the number and type of estimated vertex normals (Figure 9 (e)). Section 7 describes the mesh reduction algorithm.
3. The reduced mesh is used as the starting point for a polynomial-patch data fitting. For every triangle, we build an implicit Bernstein-Bézier patch of low degree which interpolates the vertices and vertex normals (if defined) and least-squares approximates data points in its vicinity. The algebraic patches used (cubic A-patches [4, 5, 14]) allow a simple formulation of C^1 continuity constraints between adjacent patches, and have been extended to allow the modeling of sharp features such as linear sharp edges, piecewise-planar curved creases and sharp corners (Figure 9 (f)-(h)). We detail this phase of the algorithm in Section 8.

Some of the advantages of our method with respect to existing techniques are the following: (i) Our algorithm does not require costly global optimizations, and is therefore quite suitable for practical use. (ii) The reconstructed model is in a form that can be easily used in successive analysis or modeling steps.

Results of model reconstruction obtained with our technique are presented and discussed in Section 8.5.

6 Connect-the-Dots: Inferring Topology from Vicinity

One of the most difficult problems of surface reconstruction from unorganized points is understanding how to connect the points so as to form a surface that has the same topological (e.g. number of handles) and

geometric (e.g. depressions and protrusions) characteristics of the original.

In this chapter we will formalize the problem of surface approximation and reconstruction, give a set of sufficient conditions for reconstructing a shape using alpha-shapes, and introduce an automatic method for building an interpolating triangle mesh.

6.1 Sampling and Reconstructing a 3D Object

Reconstructing the shape of an object from an unorganized “cloud” of points is in general an under-constrained problem. Consider the simple 2D reconstruction problem illustrated in Figure 10: Several solutions are possible, and it is difficult to identify a “best” solution to the problem. It is therefore of interest looking at the following problem: What are the characteristics of a sampling S (a finite set of points) of the surface of a solid object M , such that M can be reconstructed from S unambiguously and within predefined approximation bounds?

In particular, we consider the following *reconstruction problem*: Starting with a sampling of the surface B of a solid, we want to compute a triangulated surface K that has the “same shape” of B , and such that a suitably defined *distance* $D(K, B)$ of K from B is bounded by a given ε . A useful distance measure is for example:

$$D(K, B) = \max_{p \in [K]} \min_{q \in B} \|p - q\|.$$

Stated formally:

Problem 6.1 *Let B be the boundary of a solid M , and $S \subset B$ a finite set of points (sampling). Construct a (geometric) simplicial complex K , such that $K^{(0)} = S$, K is homeomorphic to B , and $D(K, B) < \varepsilon$.*

An algorithm aimed at reconstructing the shape of an object from point data alone must have a way of inferring spatial relationships among points. Characteristics of the sampling that guarantee an unambiguous and correct reconstruction depend on how the data is interpreted by the algorithm.

We have already mentioned that alpha-shapes allow us to find spatial relationships between points of an unorganized set. The relationships are based on proximity. Clusters of points close to each other are grouped to form edges, triangles and tetrahedra, and more complex structures made of collections of these simple constituents. An alpha-shape is (the underlying space of) a subcomplex of the Delaunay triangulation (regular triangulation in the case of weighted points). Therefore, it is relatively easy to compute.

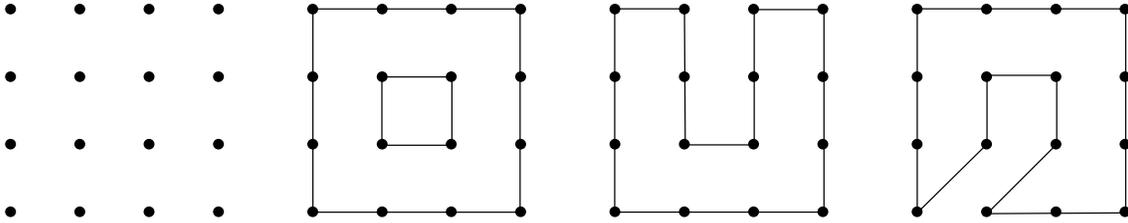


FIGURE 10: An example of ambiguous 2D reconstruction from points. From left to right: A point sampling and three, equally acceptable, reconstructions.

Let us recall that (see Section 4) an alpha-shape is a polytope whose boundary is composed of α -exposed k -simplices (convex combinations of subsets T of a point set $S \subset \mathbf{R}^3$, $k = 0, 1, 2$). A simplex σ_T is α -exposed if there exists a ball of radius $\sqrt{\alpha}$ that “touches” its vertices and does not contain any other point of S (we will look at the case where weights can be assigned to points later). If a 2-simplex σ_T is α -exposed on both sides then σ_T does not bound interior points of the alpha-shape.

Our reconstruction problem can be reformulated as follows: What are sufficient conditions of a sampling that guarantee that there exists an α such that the corresponding α -shape satisfies the requirements of Problem 6.1?

We can look at the two-dimensional case to get some insight into the problem. Figure 11 illustrates the discussion that follows. In this case, we are sampling a 1-manifold B (B is a collection of “loops”). Intuitively, we can think of the points of the sampling as “pins” that we fix on B . We now use a disk probe of radius $\rho = \sqrt{\alpha}$ to “sense” the manifold. The probe must be able to move from point to point of S on the surface, touching pairs of points in sequence, and without touching other parts of B . The pairs of points will be connected by segments of the alpha-shape, and will form loops homeomorphic (and geometrically close) to each component of B .

Clearly, a necessary condition is that no two adjacent points of the sampling are farther away than the diameter of our disk-probe, because otherwise the probe would “fall” inside the boundary of our solid object. We also need to make sure that all, and only, the edges connecting pairs of adjacent points are α -exposed. To do this, our probe needs to be small enough to be able to isolate a neighborhood of a point p on B , or, equivalently, discern “adjacent” points on B from points that are close in the Euclidean sense but not on the surface. These requirements are formalized in the following

Theorem 6.1 *Let $B \subset \mathbf{R}^3$ be a compact 1-manifold without boundary, and $S \subset B$ a finite point set. If*

1. *For any closed ball $D_\rho \subset \mathbf{R}^3$ of radius ρ , $B \cap D_\rho$ is either (a) empty; (b) a single point p (then $p \in \text{bd}(D_\rho)$); (c) homeomorphic to a closed 2-ball I , such that $\text{int}(D_\rho) \cap B = \text{int}(I)$;*
2. *An open ball of radius ρ centered on B contains at least one point of S ,*

then the alpha-shape \mathcal{W}_α of S , $\alpha = \rho^2$ is homeomorphic to B and

$$D(\mathcal{W}_\alpha, B) = \max_{p \in \mathcal{W}_\alpha} \min_{q \in B} \|p - q\| < \rho.$$

The proofs of the 2D and 3D versions of this theorem appear in [9, 10], respectively.

Notice that locally the error bound can be made arbitrarily small. In fact, for each segment $\sigma_T, T = \{p, q\}$, if $\|p - q\| = 2d$, the maximum local error is

$$\delta < \rho - \sqrt{\rho^2 - d^2}$$

which has limit zero as d tends to zero. Therefore, while a ρ -dense sampling will suffice to reconstruct the manifold B with distance bounded by ρ , we can always make the approximation error arbitrarily small in any region $C \subseteq B$ by simply sampling C at a higher density. Also note that the expression for δ converges to zero quadratically, that is it is sufficient to double the density of the sampling to reduce the error by a factor of four.

The conditions above restrict the domain of applicability of our reconstruction tool to curves whose radius of curvature is larger than ρ , as otherwise the ball-intersection requirement is impossible to satisfy (see Figure 12). Note however the following: (i) This restriction parallels the band-limited requirement in Nyquist’s theorem; (ii) ρ can be made (at least in theory) arbitrarily small. The price to pay to reconstruct small-scale features is to use a high-density sampling,

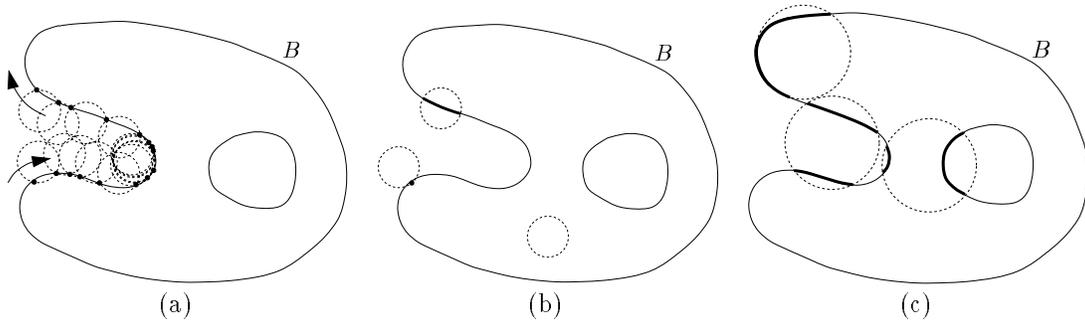


FIGURE 11: Sampling requirements for 1-manifolds in \mathbf{R}^2 . (a) The sampling density must be such that the center of the “disk probe” is not allowed to cross B without touching a sample point. (b) The radius ρ of the disk probe must be small enough that the intersection with B has at most one connected component. (c) Examples of non admissible cases of probe-manifold intersections.

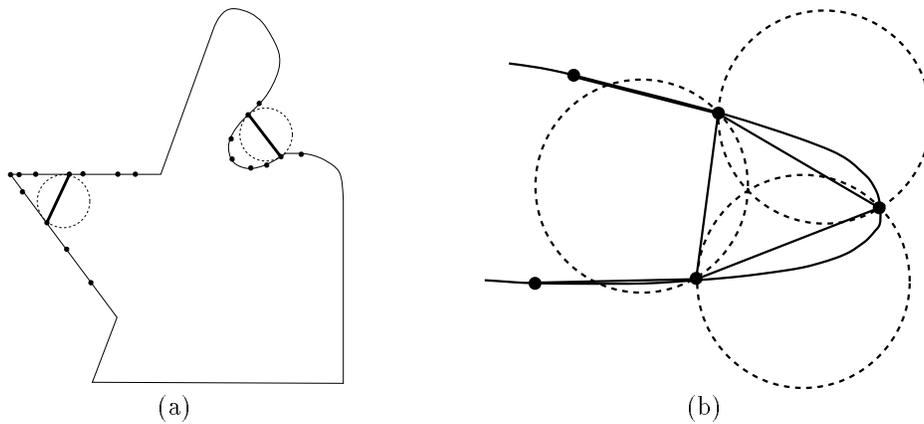


FIGURE 12: A small neighborhood of regions of curvature higher than ρ can be incorrectly reconstructed by the alpha-shape \mathcal{W}_{ρ^2} . Bold segments represent “extraneous” alpha-exposed 1-simplices. (a) A convex sharp feature and a concave high-curvature feature. (b) Extraneous alpha-exposed 1-simplex (detail).

which is reasonable. On a more practical side: (iii) the sampling density of laser scanners is usually much smaller than typical object features (otherwise large measurement errors would occur), and (iv) data collected in proximity of sharp (or high-curvature) features is subject to noise, and therefore not reliable. Accurately reconstructing sharp features (for example to segment the surface into a collection of smooth faces) requires an elaborate analysis of the data, additional knowledge of surface characteristics, and some art (see recent review [53]). We will describe later how to deal in practice with small features not directly captured by the alpha-shape.

Note also that one can use weighted alpha-shapes to reconstruct objects that have been sampled at multiple resolutions. For example, one might scan some parts of an object at a relatively coarse resolution, and other, more complicated, parts at a finer resolution, and assign different weights to the points of each scan (an example is shown in Figure 18).

6.2 Alpha-solids

While the theorems above give us sufficient conditions for a sampling to allow a faithful reconstruction using α -shapes, in practice one has to deal with less than ideal scans.

In general, i.e. when the conditions of the theorems above are not satisfied, an alpha-shape is a non-connected, mixed-dimension polytope. We are interested in reconstructing *solids*, and therefore it is convenient to define a “regularized” version of an alpha-shape. The regularization should eliminate dangling and isolated faces, edges, and points from the alpha-shape, and recognize solid components. We will define the alpha-solid by giving a construction algorithm for it.

Definition 6.1 *Assume that the regular triangulation and family of alpha-shapes of S has been computed. Then we start from a tetrahedron with a vertex “at infinity”, mark it as exterior, and proceed to visit adjacent tetrahedra, without crossing faces that belong to the alpha-shape. All tetrahedra we can reach in this way are marked as exterior. Now consider the boundary of the set of marked tetrahedra. If it is not empty, it is formed by one or more continuous shells of triangles. We start another search from all unmarked tetrahedra that have a face on this boundary, and mark all tetrahedra that can be reached without traversing alpha-shape faces as internal. We repeat this procedure until all tetrahedra have been marked.*

The union of all interior tetrahedra will be called the alpha-solid \mathcal{S}_α of S .

The alpha-solid is clearly a homogeneously three-dimensional object. Observe also that it can be computed very efficiently from the underlying triangulation, by simply traversing the adjacency graph.

Varying α , one obtains a finite collection of different alpha-solids, ranging from the empty set for sufficiently small values of α , to the convex hull of the set of points for α large enough.

Since we know that the point sampling comes from a two-manifold, we can search automatically for the “best” approximating alpha-solid. Because alpha-solids can be ordered with respect to the parameter α , and there is only a finite number of different alpha-solids (and corresponding α -values), we can perform a binary search on the family of alpha-solids. For example, to reconstruct a single connected body, we want the following properties to be satisfied by the alpha-solid:

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

To search for the minimum α such that the corresponding alpha-solid satisfies these requirements we proceed as follows:

1. Compute the regular triangulation \mathcal{T} and threshold values for alpha-shapes. Let k be the number of different α values.
2. Do a binary search on the value of α .
 - (a) Start with $\mathbf{min} = 0$, $\mathbf{max} = k - 1$
 - (b) Set $\mathbf{mid} = \lfloor (\mathbf{min} + \mathbf{max})/2 \rfloor$ and $\alpha = \alpha_{\mathbf{mid}}$. Compute the corresponding α -solid.
 - (c) Check whether the α -solid satisfies the three properties above. If it does, then try a smaller value of α : Set $\mathbf{max} = \mathbf{mid}$ and go back to step 2b. If it doesn't, then try a larger value of α : Set $\mathbf{min} = \mathbf{mid} + 1$ and go back to step 2b. If $\mathbf{min} \geq \mathbf{max}$ then exit the loop.

This algorithm takes $O(\log n)$ time (where n is the size of S) because the number of possible α -values is bounded by the number of simplices in \mathcal{T} , which is polynomial in n . Observe also that the convex hull satisfies the three properties above, so the algorithm always terminates successfully. For a sufficiently dense and uniform sampling of the object boundary the α -solid selected by this strategy is a good approximation of the object's shape. However,

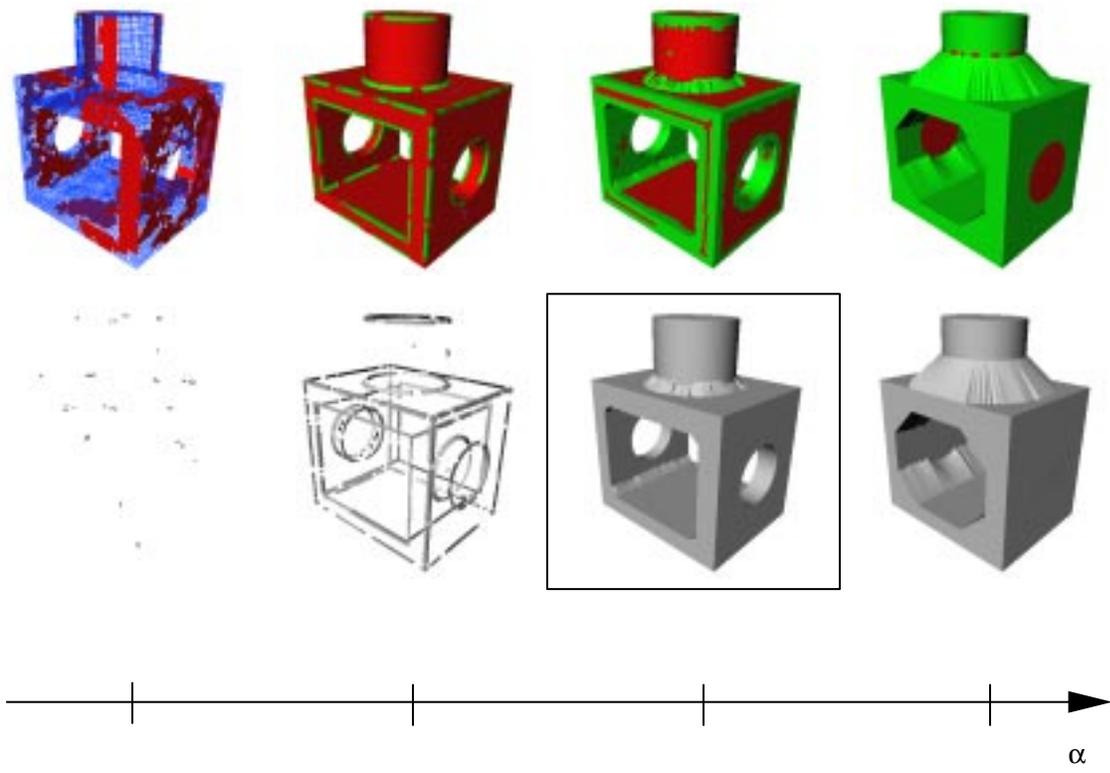


FIGURE 13: Automatic selection of an optimal α -value.

small, concave features may be occluded by unwanted tetrahedra, and some of the sampled points might lie in the interior of the α -solid (see Figure 14). We improve the initial alpha-solid with the technique described in the following section.

6.3 Improving the Alpha-Solid

Our criterion for improving the initial alpha-solid is based on the search for a subset of tetrahedra whose boundary interpolates all the points, and that maximizes the “smoothness” of the triangle mesh, defined as

$$\sum_{e_i} |\pi - \gamma_i|$$

over all edges of the triangle mesh, where γ_i is the dihedral angle formed by the two triangles incident on edge e_i .

Searching for the global optimum for this optimization problem would be clearly computationally expensive (if not intractable: This problem is probably not easier than computing a mesh of minimal surface area, which has been conjectured to be NP-complete).

However, in practice our alpha-solid is already a good approximation of the optimal polyhedron, and we only need to modify it where concave, high curvature features are present. We therefore resort to a simple greedy strategy, similar to the “sculpturing” approach proposed by Boissonnat [12] (see also Section 3.1). However, we apply the iterative removal of tetrahedra only to locally improve the alpha-solid, rather than as a global strategy to extract an interpolating mesh from the 3D Delaunay triangulation. Our technique, based on the smoothness of the mesh, gives, in our experience, better results, especially when sharp features are present (see Figure 14).

The greedy optimization algorithm is as follows. All tetrahedra having one or more boundary faces are inserted in a priority queue, where the max priority is given to tetrahedra with the largest value of the max distance between a boundary face and their circumscribing sphere. These candidate tetrahedra are then extracted from the queue one by one and considered for removal.

A candidate tetrahedron is removed if and only if:

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

When only these two types of tetrahedra are removed, the boundary of the remaining set of tetrahedra is guaranteed to remain a manifold [12]. After a candidate tetrahedron has been removed, adjacent tetrahedra whose faces become boundary faces are inserted in the priority queue as candidates for removal.

The smoothness criterion mentioned above is here stated more precisely:

Definition 6.2 Consider a tetrahedron having exactly two boundary faces (see figure 15). These faces form a dihedral angle γ_0 . They also form four dihedral angles $\gamma_1 \dots \gamma_4$ with adjacent boundary faces. Let \mathcal{S} be the following sum:

$$\mathcal{S} = \sum_{i=0}^4 |\pi - \gamma_i|$$

\mathcal{S} gives a measure of the “local smoothness” of the mesh: If the dihedral angles formed by all adjacent boundary faces are all close to straight angles, then \mathcal{S} is small.

Assuming that the tetrahedron is removed, its two internal faces become boundary faces. We can measure what the new local smoothness, say Λ , would be, and compare it with \mathcal{S} .

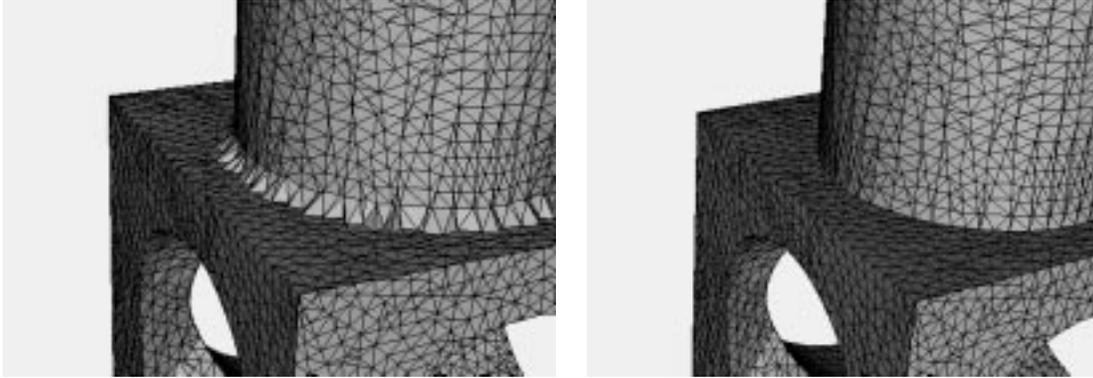
The local smoothness criterion is therefore the following: remove the tetrahedron if and only if $\mathcal{S} > \Lambda$, that is, if the local smoothness improves.

As in all greedy optimization strategies, the algorithm above might get trapped in a local minimum. For example, at some point it might become impossible to remove any tetrahedron, because all have two boundary faces and none satisfies the smoothness criterion. We therefore do a “look-ahead” search before deciding whether to remove a tetrahedron that does not satisfy the criterion: If by removing it, and some other tetrahedra adjacent to it that consequently become candidates, the smoothness criterion is satisfied, then we remove it. The depth of such look-ahead search can be limited, for all practical purposes, to a small integer value (we have used 10 for all our examples).

Figures 16, 17 and 18 and Table 1 illustrate some examples of alpha-solids computed with the technique described above.

7 Mesh Simplification

Surface mesh simplification refers to a general category of techniques designed to generate compact, adaptive approximations of dense tessellated surfaces. Optimization methods [51, 33, 30] have produced good results, but their time-intensive nature



(a)

(b)

FIGURE 14: Sculpturing to locally improve an α -solid. (a) Initial approximation.(b) After sculpturing.

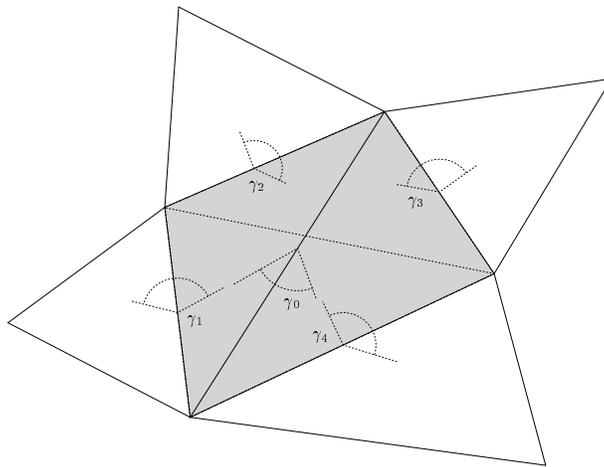


FIGURE 15: Local smoothness criterion.

<i>Object</i>	<i>Number of Points</i>	α - <i>Solid Time</i>	<i>Number of tetrahedra</i>	<i>Removed tetrahedra</i>	<i>Number of Triangles</i>
Femur	9807	1.5	36182	3704	19610
Tibia	9200	1.4	33232	2172	18396
Fibula	8146	1.1	30876	2896	16288
Patella	2050	0.3	7536	683	4096
Part 1	13040	2.5	42507	2473	26088
Club	16864	4.1	58657	754	33142
3 Tori	10833	2.2	42970	2914	21692
Bunny	33123	19.6	127607	3761	66224
Mannequin	10392	2.1	35383	2077	19802

TABLE 1: Results of alpha-solid reconstruction. The table show 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 tetrahedra in the initial alpha-solid; (4) The number of tetrahedra removed by the heuristic; (5) The number of triangles in the boundary of the final reconstructed model.

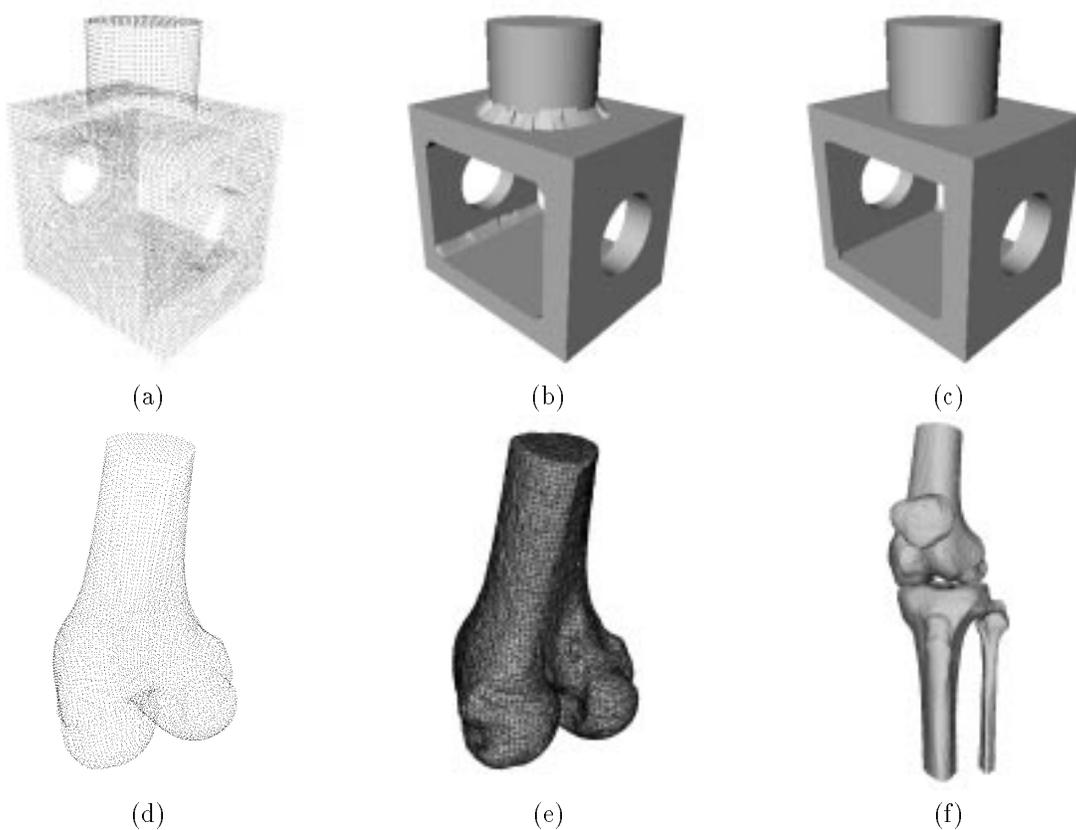


FIGURE 16: Examples of alpha-solids. (a)-(c) The data points are from a random sampling of a model (Part 1) created with a commercial solid modeler. (d)-(f) The knee data is from an isosurface extracted from the Visible Human Project data set. Data was reduced by eliminating roughly 50% of the original points.

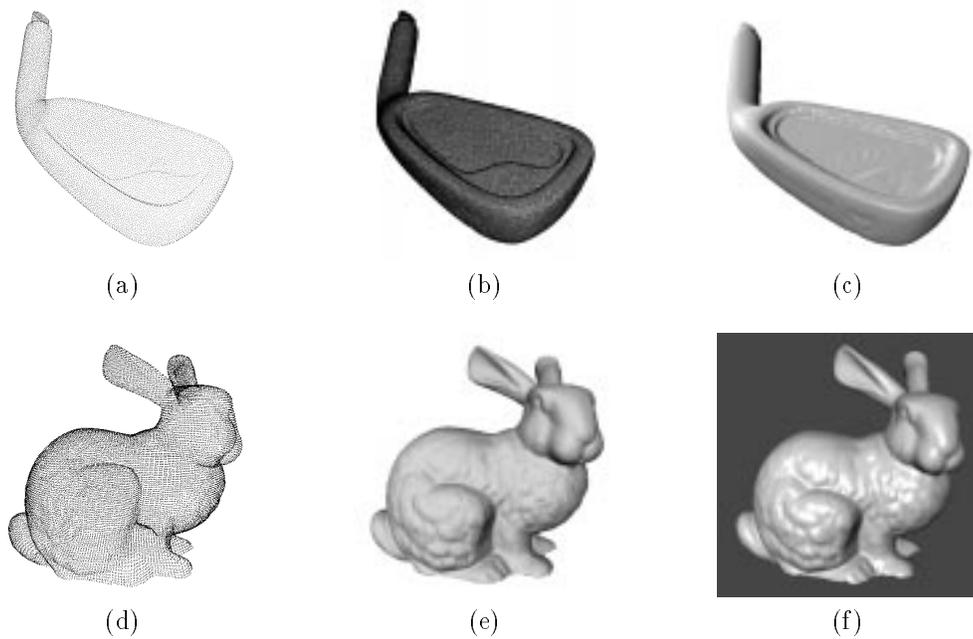


FIGURE 17: Reconstruction from range data. (a) and (d) Combined scans. (b) and (e) Reconstructed alpha-solid. (c) and (f) Phong-shaded rendering.

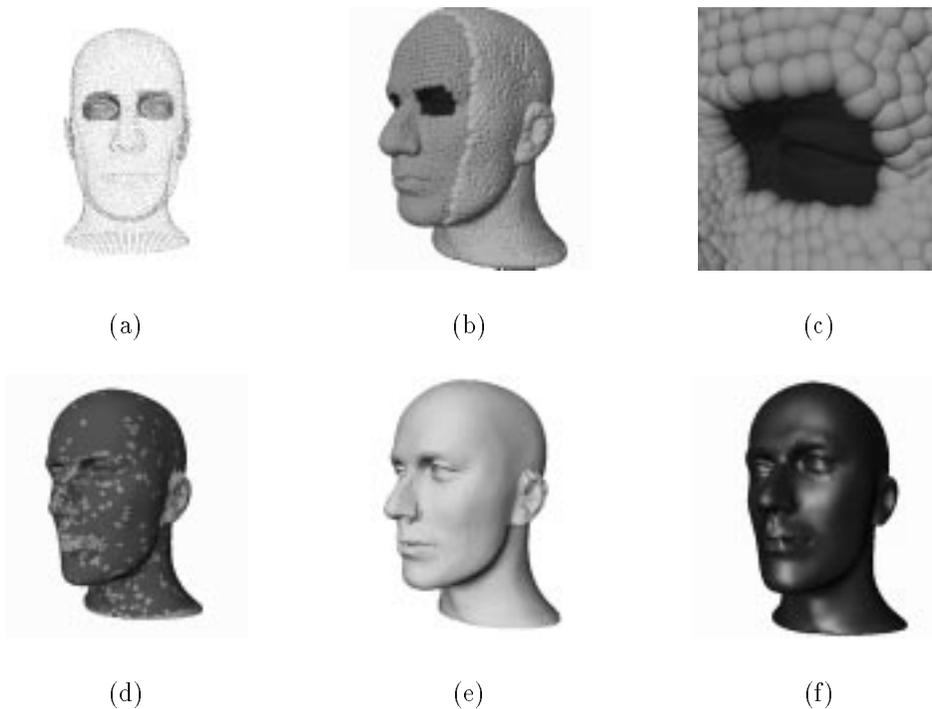


FIGURE 18: Example of reconstruction from a multi-resolution scan using weighted alpha-shapes. (a) Sampling. Notice how the eyes area has been scanned at higher resolution. (b) and (c) Weighted points represented as balls. Weights were assigned manually to simulate a multi-resolution scan. (d) Reconstructed alpha-solid. (e) Alpha-solid after improvement by sculpturing. (f) The same reconstructed model Phong-shaded.

leads us to consider more practical methods. He *et al.* [29] sample a geometric object into a volume buffer which is then low-pass filtered. Multiresolution surfaces are then extracted from the volume buffer using traditional isosurfacing techniques. Such an approach effectively generates nested approximations of surfaces, however the surfaces at each level of detail do not adapt well to large and small features simultaneously. A broad family of algorithms based on local point, edge, or triangle deletion and retriangulation based on geometric criteria are attractive for their ease of implementation, ability to capture sharp features, high simplification rates and fast performance [46, 28, 8, 16].

In this work, we adapt and improve the method of [8] to handle explicit edge feature detection and preservation. The resulting algorithm is able to maintain a strict bound on the distance between the original mesh and the surface mesh, in addition to maintaining sharp features in the reconstructed triangulation.

7.1 Mesh Simplification Algorithm

The simplification algorithm follows the basic strategy of other “vertex deletion” schemes, and is based on accumulated error bounds which are propagated from the original surface mesh through the successive simplified meshes produced by point deletion. A compact error representation consisting of two scalar error bounds per triangle is used. The error values correspond to a bound on the error (geometric displacement) toward the outside (inside) of the object. These bounds effectively form an envelope surrounding the simplified mesh which is guaranteed to contain the original surface, thus maintaining a bound on the total amount of accumulated error through successive deletion of vertices. The algorithm can be summarized as follows:

1. Initialize errors on all triangles to 0
2. Initialize priority queue P of candidate vertices v_i
 - (a) Classify v_i according to vertex configurations in figure 19
 - (b) Compute an initial triangulation of the neighbors of v_i
 - (c) Perform edge flipping to lower the error in the new triangulation
 - (d) Assign priority based on introduced error associated with v_i
3. While next candidate vertex v from P does not violate error constraints
 - (a) Delete v and incident triangles
 - (b) Add new triangles
 - (c) Update error values for new triangles
 - (d) Update P

In the following subsections, we will describe the steps of the algorithm and our extensions for sharp feature detection.

7.2 Geometric Error

Errors introduced by deletion of a vertex v are computed by establishing a mapping between the current triangulation about v and the retriangulation of the neighbors of v . This is accomplished by normal projection of the new triangulation onto the original surface, as indicated in Figure 20. As shown in the figure, this projected triangulation effectively segments the region surrounding a vertex into simple wedge shapes. In this local mapping of triangulations, the maximal error (distance between triangulations) occurs at the intersections of new edges with the original edges, and need only be computed at these points. In one exceptional case, the triangle which contains the vertex v which is being deleted, the error must also be computed between v and the point which projects to it.

Errors in the geometry are quantified by the signed distance spanned by the mapping from one triangulation to another. We use the convention that a displacement toward the *outside* (in the direction of the normal) of a mesh is a positive displacement, while displacement toward the *inside* is a negative displacement.

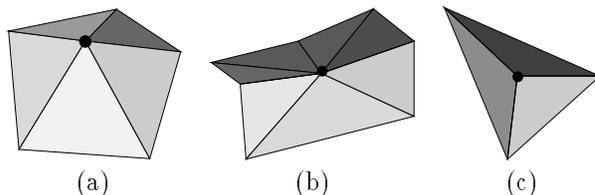


FIGURE 19: Types of candidate vertices. (a) A “smooth” vertex (all dihedral angles are larger than feature angle). (b) A vertex along a feature edge. (c) A corner.

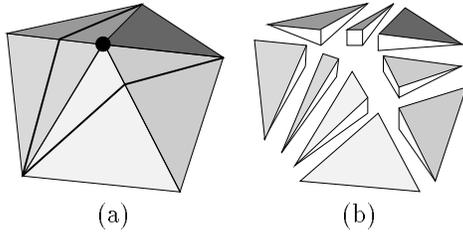


FIGURE 20: Segmentation of mutual projection. (a) Projection of new triangulation onto the old one. (b) Induced segmentation.

Positive and negative errors contribute to the *outside* and *inside* accumulated error bounds as illustrated in Figure 21. The current error bounds are represented as thin dashed lines, while the accumulated error bounds for the edge from V_i to V_j are represented by the thick dashed lines. As expected, deletion of a concave vertex requires an increase in the *inside* error bound, while deletion of a convex vertex requires an increase in the *outside* error bound.

7.3 Retriangulation

Our retriangulation strategy starts with an initial triangulation of the “hole” left by v and all adjacent triangles, and then performs edge-flippings to improve the approximation error. In order to accurately capture edge features, we modify this portion of the algorithm to detect edges in the triangulation of v by computing the dihedral angle for each pair of adjacent triangles incident to v . Edges are classified as features by a given threshold dihedral angle, as in [46]. In the case of two feature angles incident to v , our initial triangulation is constrained to contain the edge

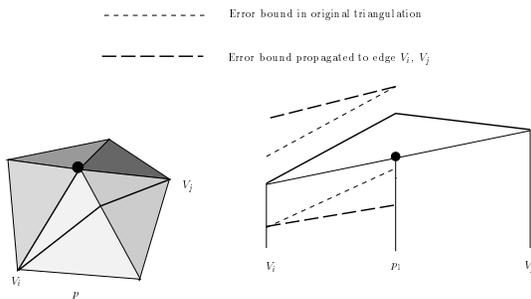


FIGURE 21: Propagating error bounds to a new triangulation.

defining the feature. If three or more feature edges are incident with v , the vertex will not be deleted.

Retriangulation proceeds by flipping interior edges which decrease the error introduced by the deletion of v . Edges are flipped only when the resulting triangulation is non self-intersecting, and no long, thin triangles are introduced. In order to maintain the computed features, edges introduced according to the dihedral angle condition will not be flipped.

7.4 Vertex ordering

The order of vertex deletions is controlled by the priority queue. Initially, a priority is assigned to each vertex by computing the error introduced through the deletion and subsequent triangulation described above. Vertices which introduce little error, and whose deletion will not violate the accumulated error bounds, are given preference over those which introduce greater error. This approach helps to ensure that our error bounds will remain relatively close to the actual error, giving a good estimation of the actual error incurred through successive deletions of points. When each point is deleted, those vertices in the local neighborhood are revisited, their priorities are recomputed and their position in the priority queue updated.

7.5 Feature Classification and Normals Estimation

We need to *tag* edges of the simplified mesh which correspond to sharp features in the data, and to estimate normals at vertices of the mesh. This information will be used in the data fitting phase.

As we said above, sharp (linear or curved) creases are detected in the initial, fine mesh by computing the dihedral angle between adjacent faces. Where three or more edges meet at a vertex we have a sharp corner. When an edge is recognized and tagged as sharp, it is never flipped during the simplification. Points between two adjacent sharp edges can however be removed, as long as the other bounds and conditions of the simplification algorithm are satisfied. Points removed along sharp features are kept in a list to be used later in the data fitting step.

For every vertex in the simplified mesh we need to estimate one or more (in the case of a vertex lying on sharp feature) normals. Some of the possible cases are depicted in Figure 22. The simplest case is that of a *smooth* vertex. None of the edges incident on the vertex is tagged as sharp, and the normals of all incident triangles form small pairwise angles. In this case a unique normal (computed by averaging

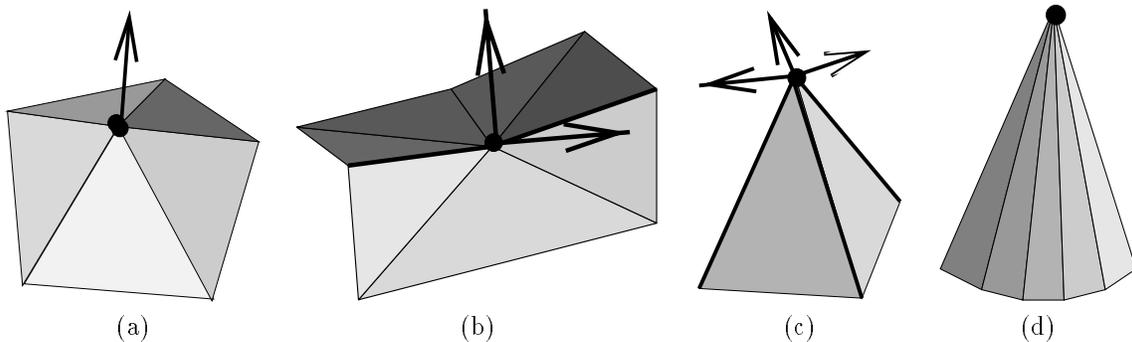


FIGURE 22: Feature classification and normal estimations: A smooth vertex in which a unique (average) normal is defined (a). Two edges along a crease (b); two distinct normals are defined at the vertex. Several sharp edges meeting at a corner (c); each sector between two sharp edges has its own normal. A singular vertex (d); no normal is defined.

all triangles normals) is associated with the vertex. Notice that even if all edges incident on a vertex are not sharp, the vertex itself can still be singular. This is for example the case of the apex of a cone. This condition is easily checked by looking at the angle formed by normals to pairs of incident faces. In this case, no normal is defined for the vertex, and the vertex is tagged as *singular*. A mix of the previous situations occurs when two or more sharp edges are incident on a vertex. These edges partition the region around the vertex into *sectors*, for each of which we will compute a separate normals. Again, the group of triangles forming a sector can have similar normals (therefore defining a smooth sector with a unique, averaged normal) or rather divergent, in which case the sector is singular and no normal is defined for it.

In summary, we can say that the region around a vertex can be divided into one or more sectors, and each sector has an associated normals (undefined in the case of a singular sector). Notice that normals can be computed from the initial fine mesh, and can therefore provide a rather good estimate of the surface curvature.

8 Piecewise-Smooth Reconstruction

Our A-patch fitting scheme interpolates the vertices (and estimated surface normals) of the simplified mesh computed as described above, and approximates the remaining data points. Features tagged as *sharp* during mesh simplification are retained in the resulting piecewise-smooth model. The fitting process begins with the construction of a tetrahe-

dral mesh to act as support for the A-patches. Then, weights for each patch are set to interpolate vertices and sharp features, and least-squares approximate the remaining point. Finally, a fairing and fitting optimization can be applied to improve the quality of the reconstructed model.

8.1 Simplicial-Hull Construction

Piecewise algebraic surfaces require a support mesh of tetrahedra, built over a base triangle mesh. Each tetrahedron contains a single, smooth (except for controlled singularities required to model C^0 features) and single sheeted A-patch. A patch passes through the three base vertices of the support tetrahedron, interpolates specified surface normals at these vertices, and approximates other data points within the tetrahedron. Figure 6 shows an example of simplicial hull, and Figure 23 illustrates the detail of the construction for a pair of adjacent triangles.

Different schemes for constructing a suitable tetrahedral mesh, called *simplicial-hull*, have been proposed in the literature (see Section 4 for related prior work). A general simplicial-hull scheme, guaranteed to satisfy conditions for the construction of a globally C^1 A-patch interpolant, is described in [5]. This scheme however does not deal with additional data points to be approximated. We briefly describe in the following the simplicial-hull construction algorithm, and a simple extension to handle approximation of internal data points.

The basic construction step consists in building, for each triangle of the base mesh, a pair of tetrahedra, on the two sides of the triangle. These *face-tetrahedra* will contain a pair of patches that interpolate the

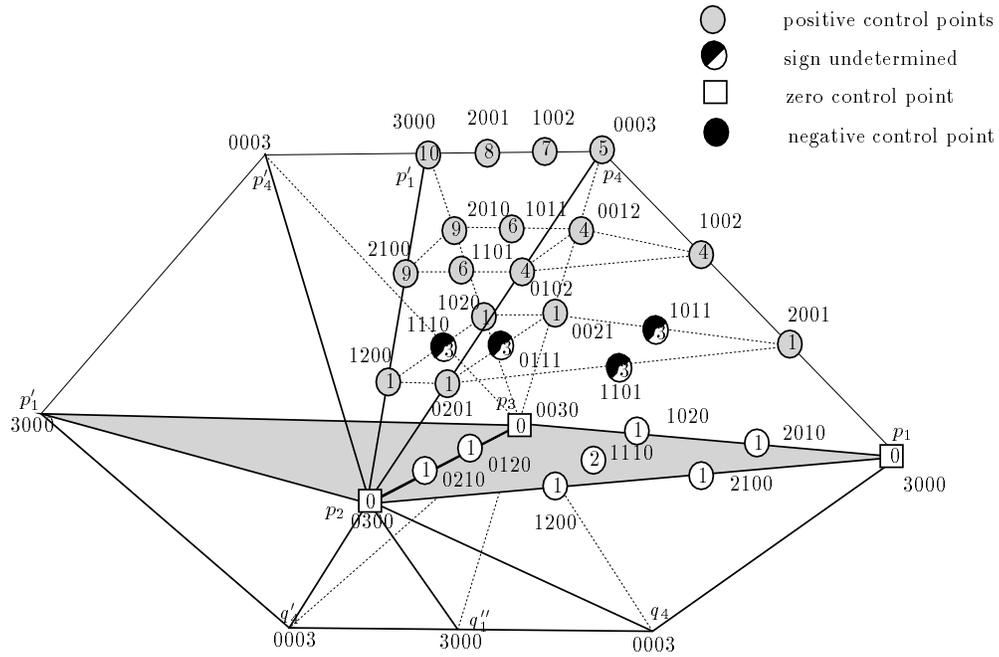


FIGURE 23: Construction of a simplicial hull. The triangles $[p_1 p_2 p_3]$ and $[p'_1 p'_2 p'_3]$ belong to the initial triangle mesh. Vertices p_4, p'_4, q_4, q'_4 have been introduced to form the four *face-tetrahedra* (on the two opposite sides of each original triangle). Vertices p'_1 and q'_1 are needed to form the four *edge-tetrahedra*. The picture also shows the assignment of weights for two patches joining with C^1 continuity.

three vertices of the base triangle (refer to Figure 23). Gaps between face-tetrahedra built on top of adjacent triangles are filled by pairs of *edge-tetrahedra*, two on each side of the base mesh. The main requirements for the simplicial hull are as follows:

Simplicial complex: The collection of tetrahedra forms a valid (geometric) simplicial complex, i.e., the only intersections occur between shared faces.

Tangent-plane containment: For each vertex v (with an assigned normal) there exist a closed neighborhood of v on the tangent plane that is contained in the collection of tetrahedra incident on v .

Point containment: The collection of tetrahedra contains all the additional data points to be approximated.

A classification of special cases is given in the cited work [5]. Mainly, one is concerned with highly convoluted triangle-meshes and wild assignments of vertex normals. Heuristics to build a simplicial hull in all these cases are given (local retriangulation can also be used to avoid some tricky situations). We should note however that such pathological cases, which must be dealt with in the general case, are unlikely to occur in our triangle meshes, as they are derived by a simplification process that uses large triangles in smooth areas, but approximates the shape of the surface with smaller triangles in areas of large curvature. Note also that, where sharp features occur, the tangent plane containment property can be relaxed.

8.2 Setting the Weights for Piecewise-Smooth Fitting

The scheme used here follows Bajaj, Chen and Xu [5], and consists basically in building two tetrahedra for each triangle of the mesh (*face-tetrahedra*) and four tetrahedra for each edge (*edge-tetrahedra*), see Figure 23. Splitting is only used in special cases.

Figure 23 also illustrates the scheme to set the weights of a patch under C^1 continuity constraints [5]. The sequence of steps involved is as follows:

1. Set weights $\textcircled{0}$ to be zero so that the surface interpolates the vertices. Set weights $\textcircled{1}$ around each vertex according to its normal so that the surface interpolates the normal. Weights $\textcircled{2}$ will be set in the subsequent least-square approximation phase.

2. Weights $\textcircled{1}$, $\textcircled{2}$ and $\textcircled{3}$ around $[p_2p_3]$ must be affine coplanar. Solve for $\textcircled{3}$ according to the others.
3. Compute weights $\textcircled{6}$ according to weights $\textcircled{4}$, $\textcircled{3}$ and $\textcircled{1}$. Preset weights $\textcircled{4}$ to be positive enough so that weights $\textcircled{6}$ are also positive.
4. Compute weights $\textcircled{7}$ according to weights $\textcircled{5}$ and $\textcircled{4}$. Preset weights $\textcircled{5}$ to be positive enough so that weights $\textcircled{7}$ are also positive.
5. Set weights $\textcircled{8}$ to be positive. Compute weights $\textcircled{9}$ by averaging $\textcircled{6}$ and $\textcircled{8}$. Weight $\textcircled{10}$ is defined by the C^1 condition.

We now extend the scheme to allow for C^0 features to be represented, and show how to use a local energy-minimization approach to set the free weights for a good data-fit. We will restrict ourselves to the following types of features (recall Figure 22):

1. Sharp corners (multiple normals defined, for example the corner of a cube).
2. Singular vertices (no normal defined, for example the apex of a cone).
3. Straight edges (two normals, one for each side, defined at each endpoint).
4. Planar or piecewise-planar curved ridges (two normals at each endpoint).

Notice that linear or planar features (straight edges or planar face) can be interpolated by simply setting all weights on the corresponding edge (or face) equal to zero.

We will now describe in some detail how to set the weights for the case illustrated in Figure 23, in which the edge between the two shaded triangles has been tagged as *sharp*, and a planar, curved sharp edge must be represented. The setting of the weights for this C^0 case is similar to the C^1 fitting described above:

1. Determine the plane that contains the curve. In the mesh simplification phase, we store points associated with a sharp feature. We use these points now to determine a best-fit plane through the edge $[p_2p_3]$. This plane determines the position of p'_1 (or p'_1).
2. Weights $\textcircled{0}$ are set to zero so that the surface interpolates the vertices.

3. Weights ① are set according to the normals at p_2 and p_3 . Let's call these normals n_2, n'_2, n_3, n'_3 . Weights ① on the two sides of $[p_2p_3p'_1]$ depend on the projection of the normals on the plane of triangle $[p_2p_3p'_1]$. Let's call these projections $\bar{n}_2, \bar{n}'_2, \bar{n}_3, \bar{n}'_3$. To achieve C^0 -continuity on face $[p_2p_3p'_1]$, all weights must be the same on its two sides (after normalization by a scaling factor). This is possible only if \bar{n}_2, \bar{n}'_2 (respectively \bar{n}_3, \bar{n}'_3) lie in the same direction, and if the ratios $|\bar{n}_2|/|\bar{n}'_2|$ and $|\bar{n}_3|/|\bar{n}'_3|$ are equal. If the condition on the normals is not satisfied, we need to "adjust" the normals (for example by averaging).
4. The following steps are similar to the C^1 case, except that obviously the C^1 conditions on weights do not propagate through face $[p_2p_3p'_1]$.
5. Weights ⑨ and ⑩ can be set freely and will be used to improve the fit to the data.

8.3 Least-Squares Approximation

We have seen in the previous section that some of the weights in each patch are free, i.e. their value can be set to guarantee a good fit to the data points.

Bajaj, Ihm and Warren [7] suggest the use of algebraic distance to measure the distance between a point \vec{p} and a surface $f(\vec{x}) = 0$. The algebraic distance between a point \vec{p} and a surface $f(\vec{x}) = 0$ is the value of $f(\vec{p})$ provided that $f(\vec{x}) = 0$ is normalized. The normalization used in the paper is $\vec{b}^T \vec{b} = 1$, where \vec{b} is the weights of $f(\vec{x})$. In the A-patch scheme, as some of the non-zero weights are given according to data at the vertices of the triangulation, no normalization is needed.

The function to be minimized can be written as follows:

$$E = \sum_i \sum_{j=0}^n (w^T B(\alpha_j))^2 = w^T Q w$$

where i is the index of an A-patch and j is the index of an auxiliary point within the domain simplex. However, some equality and inequality constraints must be added to the system to make sure that continuity and single-sheeted conditions are satisfied by the weights. We therefore obtain a quadratic programming problem, which can, in general, be expressed as

$$\begin{aligned} & \text{minimize} && \frac{1}{2} w^T Q w + w^T c \\ & \text{subject to} && a_i^T w = b_i, && i \in E \\ & && a_i^T w \leq b_i, && i \in I. \end{aligned} \quad (4)$$

where E and I are index sets for equality and inequality constraints. The matrix Q is symmetric and positive semidefinite (if not actually positive definite).

The general quadratic program with inequality constraints is almost always solved by an active set method (see [37], Section 11.3). There is an especially simple version for the case where Q is positive definite. More details on the solution of this problem can be found in [14].

8.4 Fairing and Fitting Optimization

Fairing a surface is the process of slightly modifying its shape to improve the distribution of curvature across it. Although the surfaces produced by our fitting scheme are mathematically derivative-continuous, fairing is usually required to "remove the wrinkles" and obtain surfaces that are more aesthetically pleasing. Fairing strategies are usually based on minimization of a functional that depends on the second derivatives of the surface. A popular choice is for example the thin-plate strain energy, which is proportional to the surface integral of the sum of principal curvatures squared:

$$\int_S \kappa_1^2 + \kappa_2^2 dS$$

The problem has been studied in detail for the parametric case. In general, its solution requires costly global optimization algorithms.

For implicit surfaces little is known. Chen [14] discusses two possible methodologies. The first technique is based on iterative, numerical minimization of a functional that measures the strain energy of a thin-plate spline. The second technique, simpler and faster, is formulated as the integral of a weighted sum of the squares of the first and second derivatives of each patch over its domain tetrahedron. Such an energy function is easy to be minimized as the optimization problem can be formulated as a quadratic programming problem with linear inequality constraints. The fairing process can be combined with an optimization of the fitting to the data points.

While the approaches outlined above gave promising preliminary results, more research is needed to devise an efficient and general fairing and optimization for composite algebraic surfaces. We will further investigate these issues in the future.

8.5 Results

Examples of reconstruction of 3D models are presented and discussed in this Section.



(a)



(b)



(c)



(d)



(e)



(f)

FIGURE 24: Example of an object reconstruction. The shoe was sampled at about $1 \cdot 10^4$ points (a). The selected α -solid is shown in (b). The simplified mesh, 310 triangles (c) is used in (d) to build the simplicial hull and fit the data with polynomial algebraic patches (e). The final result is shown in (f).

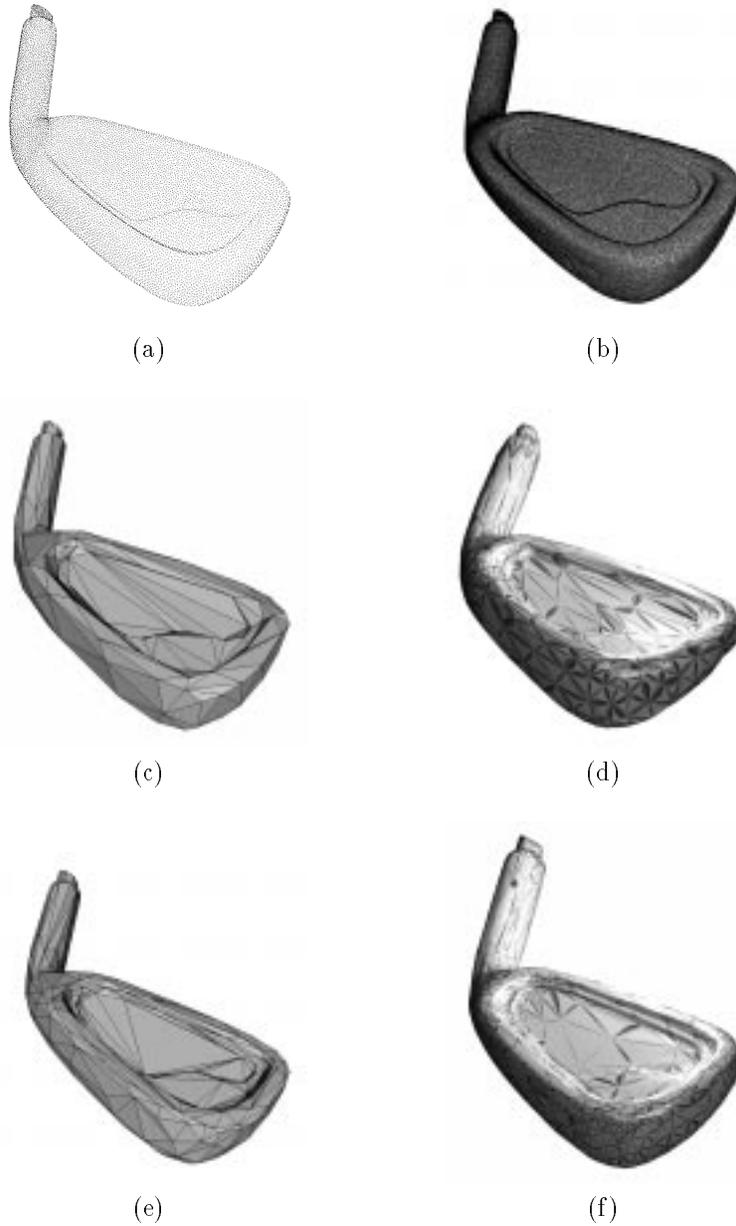


FIGURE 25: Example of reconstruction from real range data. The scan (a) contains 16000 points. The α -solid (b) is a mesh of 33142 triangles, simplified to 268 by our algorithm (c). The reconstructed object contains about 1000 patches and is shown in (d). Another reconstruction, based on a simplified mesh of 636 triangles (e), results in 2500 patches (f). Different colors have been used to distinguish between face- and edge-patches.

<i>Object</i>	<i>Number of Points</i>	<i>α-Solid \mathcal{E} Simp. Time</i>	<i>α-Solid Tri.</i>	<i>Simpl. Tri.</i>	<i>N. of Patches</i>	<i>Error</i>	<i>Fitting Time</i>
CSG	12128	2.5	24264	290	1680	< 1%	2.0
Shoe	9089	2.0	17786	248	1482	< 1%	2.7
Club	16576	3.2	33142	636	2522	< 1%	4.2
Club 2	16576	3.3	33142	268	1002	< 1.5%	2.8

TABLE 2: Results obtained with the α -solid preprocessing and A-patch piecewise-smooth fitting method. Time is in minutes. Errors are relative to the diameter (distance between farthest points) of the object.

Figure 24 contains a dense sampling of points from a high-heeled shoe. The original sampling consists of 9089 points, uniformly distributed over the surface. The reconstructed piecewise linear object contains 17786 triangles. This linear approximation is simplified to a rough approximation of 248 triangles, while maintaining C^0 features and a surface distance error of less than 1%. A total of about 1500 piecewise C^0 and C^1 cubic patches interpolate the simplified model and approximate the original set of points. The entire process required approximately 2.5 minutes for the computation of the α -solid and successive simplification, and about 2 minutes to set the weights of the polynomial patches for data fitting with the required continuity.

Figure 25 shows another example from real range data, reconstructed at two levels of resolution. The range data consists of over $1.6 \cdot 10^4$ uniformly distributed points. The reconstructed piecewise linear approximation of the surface contains 33142 triangles. Mesh reduction produces coarse approximations of 636 and 268 triangles while maintaining the sharp features of the object and approximating the surface to within a tolerance of 1% and 1.5% respectively. Approximately 2500 (1000) piecewise C^0 and C^1 cubic patches interpolate the vertices of the simplified model as well as the normals (when defined) while approximating the original points. The computation time includes less than 4 minutes for the initial reconstruction and mesh simplification, followed by approximately 4 minutes for A-patch fitting. All computations were carried out on an SGI Indigo2 (250 MHz MIPS R4400).

9 Conclusions

We presented in this paper a method to reconstruct the geometric shape of objects from a sampling of their surfaces. We established sufficient conditions

for the sampling to allow a homeomorphic and error-bounded reconstruction. The use of alpha-shapes (and alpha-solids) permitted us to use a solid mathematical framework (to formally prove the sampling conditions) and to develop efficient algorithms for practical uses. Weighted alpha-shapes can be used to deal with multi-resolution scans.

While other reconstruction techniques from unorganized points exist and can work well in practical applications, our alpha-shapes based technique is, to the best of our knowledge, the first reconstruction algorithm which guarantees a correct reconstruction when certain sampling conditions are satisfied.

We also described a method to convert the fine triangle mesh produced by the alpha-solid reconstruction into a more useful, and manageable, piecewise-polynomial surface model. Our method is based on implicit algebraic patches, or A-patches. While less popular than parametric representations, the A-patch modeling technique offers several advantages, from the closure with respect to basic modeling operations (offsetting, intersection) to the high design flexibility for a relatively low polynomial degree.

Our method is targeted to shapes that have sharp features. It starts with an error-bounded simplification of the alpha-solid triangle mesh. It creates a supporting tetrahedral mesh around the simplified mesh, and fits A-patches to the data points inside each tetrahedron.

One of the main advantages of these techniques over previous methods is that they are fast. In our current implementation, we reconstruct models from samplings of the order of 10^4 points in minutes on a general purpose, uni-processor workstation. The bottleneck of our current implementation is the amount of memory used by the incremental triangulation algorithm to store the history of split/flipped tetrahedra. A different implementation (or a different algorithm), could allow us to deal with larger point sets.

Notice that a laser range scan might be more dense

than is needed by our alpha-solid reconstruction algorithm. A simple prefiltering can be applied to the data set to reduce its size. In the following A-patch fitting phase, all the original measured points can be used to achieve the best possible reconstruction. As all of the computation involved in the A-patch fitting is done locally, the algorithms handle large data sets gracefully. We plan in the future to improve our fairing and fitting optimization techniques for the reconstructed A-patch model.

We have not addressed all of the problems that arise in developing a fully automatic reverse-engineering technique. We have already mentioned the problems associated with handling noisy measurements and incomplete information. We have not attempted to *understand* the shape of the object to build a representation that captures its functionality or manufacturing process.

While more research is needed to answer fully the needs of reverse-engineering applications, we believe that automatic reconstruction will become an increasingly viable and common alternative to manual or semi-automatic shape acquisition.

Acknowledgments. Thanks to the Computer Graphics Group, University of Washington and the Stanford University Computer Graphics Laboratory, for making some of the range data used in this paper publicly available.

References

- [1] C. Bajaj. Surface fitting with implicit algebraic surface patches. In H. Hagen, editor, *Topics in Surface Modeling*, pages 23–52. SIAM Publications, 1992.
- [2] C. Bajaj. The emergence of algebraic curves and surfaces in geometric design. In R. Martin, editor, *Directions in Geometric Computing*, pages 1–29. Information Geometers Press, 1993.
- [3] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 95, pages 109–118. ACM SIGGRAPH, 1995.
- [4] C. Bajaj, J. Chen, and G. Xu. Free form surface design with A-patches. In *Proceedings of Graphics Interface '94*, pages 174–181, Banff, Canada, 1994.
- [5] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics*, 14(2):103–133, 1995.
- [6] C. Bajaj and I. Ihm. C^1 smoothing of polyhedra with implicit algebraic splines. *Computer Graphics*, 26(2):79–88, July 1992. Proceedings of SIGGRAPH 92.
- [7] C. Bajaj, I. Ihm, and J. Warren. Higher-order interpolation and least-squares approximation using implicit algebraic surfaces. *ACM Transactions on Graphics*, 12(4):327–347, 1993.
- [8] C. Bajaj and D. Schikore. Error-bounded reduction of triangle meshes with multivariate data. In *Proceedings of SPIE Symposium on Visual Data Exploration and Analysis III*, pages 34–45. SPIE, January 1996.
- [9] F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. Submitted for publication, 1996.
- [10] F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds. In preparation, 1997.
- [11] F. Bernardini, C. Bajaj, J. Chen, and D. Schikore. Automatic reconstruction of 3D CAD models. Submitted for publication, 1996.
- [12] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984.
- [13] R. M. Bolle and B. C. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):1–13, January 1991.
- [14] Jindong D. Chen. *Interactive deformable modeling with A-patches*. PhD thesis, Department of Computer Sciences, Purdue University, May 1996.
- [15] B. K. Choi, H. Y. Shin, Y. I. Yoon, and J. W. Lee. Triangulation of scattered data in 3D space. *Computer Aided Design*, 20(5):239–248, June 1988.
- [16] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, and F. Brooks a W. Wright. Simplification envelopes. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 96, pages 119–128. ACM SIGGRAPH, 1996.
- [17] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 96, pages 303–312. ACM SIGGRAPH, 1996.
- [18] W. Dahmen. Smooth piecewise quadratic surfaces. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 181–193. Academic Press, Boston, 1989.
- [19] W. Dahmen and T.-M. Thamm-Schaar. Cubicoids: modeling and visualization. *Computer Aided Geometric Design*, 10:93–108, 1993.
- [20] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 96, pages 173–182. ACM SIGGRAPH, 1995.

- [21] M. Eck and H. Hoppe. Automatic reconstruction of B-splines surfaces of arbitrary topological type. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 96, pages 325–334. ACM SIGGRAPH, 1996.
- [22] H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois, Urbana-Champaign, IL, 1992.
- [23] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, IT-29:551–559, 1983.
- [24] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, January 1994.
- [25] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3:83–127, 1986.
- [26] B. Guo. *Modeling arbitrary smooth objects with algebraic surfaces*. PhD thesis, Computer Science, Cornell University, 1991.
- [27] B. Guo. Surface generation using implicit cubics. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 485–530. Springer-Verlag, Tokyo, 1991.
- [28] B. Hamann. A data reduction scheme for triangulated surfaces. *Comput. Aided Geom. Design*, 11:197–214, 1994.
- [29] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel based object simplification. In G. M. Nielson and D. Silver, editors, *Visualization '95 Proceedings*, pages 296–303, October 1995.
- [30] H. Hoppe. Progressive meshes. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 96, pages 99–108. ACM SIGGRAPH, 1996.
- [31] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schwitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 94, pages 295–302. ACM SIGGRAPH, 1994.
- [32] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, July 1992. Proceedings of SIGGRAPH 92.
- [33] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proc. SIGGRAPH '93*, pages 19–26, 1993.
- [34] C. W. Liao and Gérard Medioni. Surface approximation of a cloud of 3D points. *Graphical Models & Image Processing*, 57(1):67–74, January 1995.
- [35] S. Lodha. *Surface Approximation with Low Degree Patches with Multiple Representations*. PhD thesis, Computer Science, Rice University, Houston, Texas, 1992.
- [36] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics. University of Utah, August 1987.
- [37] D. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley, Reading, Massachusetts, second edition, 1984.
- [38] D. Moore and J. Warren. Approximation of dense scattered data using algebraic surfaces. In V. Milutinovic and B. D. Shriver, editors, *Proceedings of the 24th annual Hawaii International Conference on System Sciences*, volume 1, 1991.
- [39] J. O'Rourke. Polyhedra of minimal area as 3D object models. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 664–666, 1981.
- [40] A. Pentland and S. E. Sclaroff. Closed form solutions for physically based shape modelling and recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):715–729, 1991.
- [41] J. Peters. Constructing C^1 surfaces of arbitrary topology using biquadratic and bicubic splines. In N. Sapidis, editor, *Designing fair curves and surfaces*, pages 277–293. SIAM, 1994.
- [42] R. Poli, G. Coppini, and G. Valli. Recovery of 3D closed surfaces from sparse data. *Computer Vision, Graphics and Image Processing*, 60(1):1–25, July 1994.
- [43] D. Ruprecht and H. Muller. Free form deformation with scattered data interpolation methods. In H. Hagen G, Farin and H. Noltemeier, editors, *Geometric Modelling (Computing Suppl. 8)*, pages 267–281. Springer-Verlag, Wien, 1993.
- [44] N. S. Sapidis and P. J. Besl. Direct construction of polynomial surfaces from dense range images through region growing. *ACM Transactions on Graphics*, 14(2):171–200, April 1995.
- [45] F. Schmitt, B. A. Barsky, and W. Du. An adaptive subdivision method for surface fitting from sampled data. *Computer Graphics*, 20(4):179–188, 1986. Proceedings of SIGGRAPH 86.
- [46] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Comput. Graph.*, 26(2):65–70, 1992. Proc. SIGGRAPH '92.
- [47] T. Sederberg. Techniques for cubic algebraic surfaces, part one. *IEEE Computer Graphics & Applications*, 10(4):14–25, July 1990.
- [48] T. Sederberg. Techniques for cubic algebraic surfaces, part two. *IEEE Computer Graphics & Applications*, 10(6):12–21, September 1990.
- [49] T. W. Sederberg. Piecewise algebraic surface patches. *Computer Aided Geometric Design*, 2:53–59, 1985.
- [50] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: recovering 3D shape and non-rigid motion. *Artificial Intelligence*, 36:91–123, 1988.

- [51] G. Turk. Re-tiling polygonal surfaces. *Comput. Graphics*, 26(2):55–64, 1992. Proc. SIGGRAPH '92.
- [52] G. Turk and M. Levoy. Zippered polygonal meshes from range images. In *Computer Graphics Proceedings*, Annual Conference Series. Proceedings of SIGGRAPH 94, pages 311–318. ACM SIGGRAPH, 1994.
- [53] T. Váradi, R. R. Martin, and J. Cox. Reverse engineering of geometric models - An introduction. *Computer Aided Design*, 27(1), 1996. To appear.
- [54] R. C. Veltkamp. *Closed object boundaries from scattered points*. PhD thesis, Center for Mathematics and Computer Science, Amsterdam, 1992.