Piecewise Smooth Surface Reconstruction

Hugues Hoppe* Tony DeRose* Tom Duchamp† Mark Halstead‡
Hubert Jin§ John McDonald§ Jean Schweitzer* Werner Stuetzle§

University of Washington
Seattle, WA 98195

Abstract

We present a general method for automatic reconstruction of accurate, concise, piecewise smooth surface models from scattered range data. The method can be used in a variety of applications such as reverse engineering — the automatic generation of CAD models from physical objects. Novel aspects of the method are its ability to model surfaces of arbitrary topological type and to recover sharp features such as creases and corners. The method has proven to be effective, as demonstrated by a number of examples using both simulated and real data.

A key ingredient in the method, and a principal contribution of this paper, is the introduction of a new class of piecewise smooth surface representations based on subdivision. These surfaces have a number of properties that make them ideal for use in surface reconstruction: they are simple to implement, they can model sharp features concisely, and they can be fit to scattered range data using an unconstrained optimization procedure.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling. - surfaces and object representations; J.6 [Computer-Aided Engineering]: Computer-Aided Design (CAD); G.1.2 [Approximation]: Spline Approximation.

Additional Keywords: Geometric modeling, surface fitting, shape recovery, range data analysis, subdivision surfaces.

1 Introduction

In this paper, we present a new representation for piecewise smooth surfaces of arbitrary topological type,[1] and a method for fitting such surface models to scattered range data, where neither the topological type of the surface, its geometry, nor the location of its sharp features are known in advance. We also present examples showing that the surface representation and fitting method are useful for important applications such as *reverse engineering* — the automatic generation of CAD models from laser range data.

In previous work [4, 10, 11], we developed a method for fitting compact, accurate piecewise linear surfaces to scattered range data. The generalization to piecewise smooth surfaces is a natural and necessary extension. Many objects of interest are piecewise smooth; their surfaces consist of smoothly curved regions that meet along sharp curves and at sharp corners. Modeling such objects as piecewise linear surfaces requires a large number of triangles, whereas curved surface models can provide both a more accurate and a more compact representation of the true surface. It is critical, however, to use a surface representation that is capable of explicitly modeling sharp features. Using an everywhere smooth surface representation to model sharp features either results in a large number of surface elements, or in a poor geometric fit, as illustrated in Color Plate 1m. Additionally, the surface representation should be capable of modeling surfaces of arbitrary topological type.

The most popular smooth surface representations are tensor product NURBS. However, NURBS can only represent surfaces of arbitrary topological type by partitioning the model into a collection of individual NURBS patches. Adjacent patches must then be explicitly stitched together using geometric continuity conditions [6]. A large number of parameters (the B-spline coefficients) are therefore introduced, most of which are constrained by the continuity conditions. As a consequence, fitting NURBS in general requires high-dimensional constrained optimization.

Subdivision surfaces, first introduced by Doo/Sabin [5] and Catmull/Clark [3], offer a promising alternative. They are capable of modeling everywhere smooth surfaces of arbitrary topological type using a small number of unconstrained parameters.

Our surface representation is a generalization of the subdivision surface scheme introduced by Loop [13]. Loop's scheme, like all subdivision schemes to date, produces tangent plane continuous surfaces of arbitrary topological type. A principal contribution of our work is to show that it is possible to locally modify Loop's subdivision rules to model sharp features such as creases and corners. The modified subdivision rules also model boundary curves, as shown for instance in the spout of the Utah teapot (Color Plate 2).

Our reconstruction method consists of three major phases, the first two of which have been described elsewhere:

1. Estimation of topological type [10]: Given an unorganized set of points (Color Plate 1j) on or near some unknown surface (Color Plate 1i), phase 1 constructs a triangular mesh consisting of a relatively large number of triangles (Color Plate 1k). This phase determines the topological type of the surface and produces an initial estimate of the geometry.

2. Mesh optimization [4, 11]: Starting with the output of phase 1, phase 2 reduces the number of triangles and improves the fit to

*Department of Computer Science and Engineering, FR-35
†Department of Mathematics, GN-50
‡Apple Computer
§Department of Statistics, GN-22

This work was supported in part by IBM, Silicon Graphics Inc., the Xerox Corporation, and the National Science Foundation under grants CCR-8957323 and DMS-9103002.

[1] The topological type of a surface refers to its genus, the presence of boundaries, etc.

the data (Color Plate 1l). Our approach to this phase is to cast the problem as optimization of an energy function that explicitly models the trade-off between the competing goals of concise representation and good fit. The free variables in the optimization procedure are the number of vertices in the mesh, their connectivity, and their positions.

3. Piecewise smooth surface optimization: Phase 3 is the subject of this paper. Starting with the optimized mesh (a piecewise linear surface) produced in phase 2, this phase fits an accurate, concise piecewise smooth subdivision surface (Color Plate 1o), again by optimizing an energy function that trades off conciseness and fit to the data. The phase 3 optimization varies the number of vertices in the control mesh, their connectivity, their positions, and the number and locations of sharp features. The automatic detection and recovery of sharp features in the surface is an essential part of phase 3. Our piecewise smooth subdivision surface scheme is introduced in Section 3. The optimization problem and algorithm are described in Section 4.

Phase 2 can in principle be eliminated, but has proven to be convenient for two reasons: first, is it computationally more efficient to optimize over a piecewise linear surface in the early stages of optimization, and second, initial estimates of sharp features are much more robust when obtained from the phase 2 mesh.

The principal evidence for our method's success is its application to a wide variety of data, including simulated and real laser scanner data. A number of examples, shown in the Color Plates, are discussed in Section 5.

2 Background on subdivision surfaces

A subdivision surface is defined by repeatedly refining an initial control mesh as indicated in Color Plates 1a–1d. (Formally, a *mesh* M is a pair (K, V), where: K is a *simplicial complex* specifying the connectivity of the vertices, edges, and faces, and thus determining the topological type of the mesh; $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$, $\mathbf{v}_i \in \mathbf{R}^3$ is a set of vertex positions defining the shape of the mesh in $\mathbf{R}^3$.) The first and most popular subdivision surface schemes, introduced by Doo/Sabin [5] and Catmull/Clark [3], are based on quadrilateral meshes, and generalize biquadratic and bicubic tensor product B-splines, respectively. A subdivision scheme based on triangles is more convenient for our purposes. We use a generalization of the triangular scheme introduced by Loop [13], as it is the simplest known scheme leading to tangent plane smooth surfaces.

2.1 Loop's subdivision: Loop's subdivision scheme is a generalization of C^2 quartic triangular B-splines. The refinement step proceeds by splitting each triangular face into four subfaces. The vertices of the refined mesh are then positioned using weighted averages of the vertices in the unrefined mesh. Formally, starting with the initial control mesh $M = M^0$, each subdivision step carries a mesh $M^r = (K^r, V^r)$ into a refined mesh $M^{r+1} = (K^{r+1}, V^{r+1})$ where the vertices V^{r+1} are computed as affine combinations of the vertices of V^r. Some of the vertices of V^{r+1} naturally correspond to vertices of V^r — these are called *vertex points*; the remaining vertices in V^{r+1} correspond to edges of the mesh M^r — these are called *edge points*. Let $\mathbf{v}^r$ denote a vertex of V^r having neighbors $\mathbf{v}_1^r, \ldots, \mathbf{v}_n^r$ as shown in Figure 1a. Such a vertex is said to have valence n. Let $\mathbf{v}_i^{r+1}$ denote the edge point of V^{r+1} corresponding to the edge $\mathbf{v}^r\mathbf{v}_i^r$, and let $\mathbf{v}^{r+1}$ be the vertex point of V^{r+1} associated with $\mathbf{v}^r$. The positions of $\mathbf{v}^{r+1}$ and $\mathbf{v}_i^{r+1}$ are computed according to the subdivision rules

$$\begin{aligned} \mathbf{v}^{r+1} &= \frac{\alpha(n)\mathbf{v}^r + \mathbf{v}_1^r + \cdots + \mathbf{v}_n^r}{\alpha(n) + n} \\ \mathbf{v}_i^{r+1} &= \frac{3\mathbf{v}^r + 3\mathbf{v}_i^r + \mathbf{v}_{i-1}^r + \mathbf{v}_{i+1}^r}{8}, \quad i = 1, \ldots, n \end{aligned} \qquad (1)$$

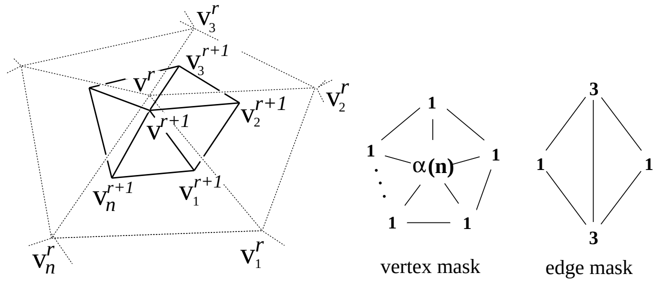

Figure 1: The neighborhood around a vertex $\mathbf{v}^r$ of valence n (left); Loop's vertex mask (center); Loop's edge mask (right).

where subscripts are taken modulo n, and where $\alpha(n) = \frac{n(1-a(n))}{a(n)}$ with $a(n) = \frac{5}{8} - \frac{(3+2\cos(2\pi/n))^2}{64}$. Affine combinations such as those in Equation 1 can be nicely visualized by diagrams called *masks*, as shown in Figure 1.

2.2 Computing surface points and tangent vectors: Loop's surfaces in particular, and subdivision surfaces in general, are defined only as the limit of an infinite refinement process. In most cases closed form expressions for the limit surfaces are not known, but somewhat surprisingly, various properties of subdivision surfaces, such as exact points on the surface and exact tangent planes, can nonetheless be computed.

To study the properties of subdivision surfaces, it is convenient to write Equation 1 in matrix form as

$$\begin{aligned} (\mathbf{v}^{r+1}, \mathbf{v}_1^{r+1}, \ldots, \mathbf{v}_n^{r+1})^T &= S_n(\mathbf{v}^r, \mathbf{v}_1^r, \ldots, \mathbf{v}_n^r)^T \\ &= S_n^{r+1}(\mathbf{v}^0, \mathbf{v}_1^0, \ldots, \mathbf{v}_n^0)^T \end{aligned} \qquad (2)$$

where superscript T denotes matrix transpose. The matrix S_n is called the *local subdivision matrix* [5].

As $r \to \infty$, each point $\mathbf{v}^r$ approaches a point on the limit surface. Equation 2 suggests that the limit point can be obtained by analyzing the eigenstructure of the local subdivision matrix. Indeed, the limit point can be expressed as an affine combination of the initial vertex positions [8]:

$$\mathbf{v}^\infty = \frac{\ell_1\mathbf{v}^0 + \ell_2\mathbf{v}_1^0 + \cdots \ell_{n+1}\mathbf{v}_n^0}{\ell_1 + \ell_2 + \cdots \ell_{n+1}}$$

where $(\ell_1, \ldots, \ell_{n+1})$ is the dominant left eigenvector of S_n. For Loop's surfaces this affine combination can be expressed as the *position mask* shown in Figure 2 [13].

Eigenanalysis of the local subdivision matrix can also be used to establish smoothness. It can be shown, for instance, that Loop's surfaces are indeed tangent plane continuous [13, 19]. Moreover, Halstead *et al.* [8] show that the tangent vectors to the limit surface at $\mathbf{v}^\infty$ can be computed using the two left eigenvectors of S_n corresponding to the second largest eigenvalue (this eigenvalue has multiplicity 2). For Loop's surfaces the vectors

$$\begin{aligned} \mathbf{u}_1 &= c_1\mathbf{v}_1^0 + c_2\mathbf{v}_2^0 + \cdots + c_n\mathbf{v}_n^0 \\ \mathbf{u}_2 &= c_2\mathbf{v}_1^0 + c_3\mathbf{v}_2^0 + \cdots + c_1\mathbf{v}_n^0, \end{aligned} \qquad (3)$$

with $c_i = \cos(2\pi i/n)$ span the tangent plane of the surface. Their cross product therefore gives an exact normal vector to the surface which is useful, for example, to create Phong-shaded renderings such as those shown in the Color Plates. The formulas given in Equation 3 can be visualized as the *tangent masks* shown in Figure 2.

Eigenanalysis will again be used in Section 3.2 to study the properties of piecewise smooth subdivision surfaces.

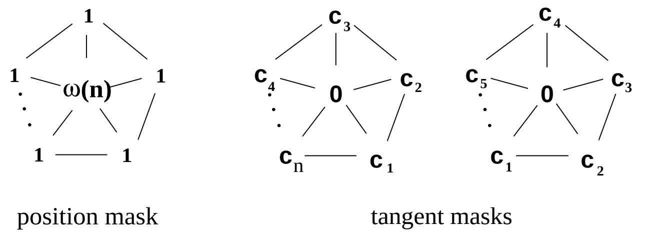

Figure 2: Masks for computing positions and tangents to Loop's surfaces, where $\omega(n) = \frac{3n}{8\alpha(n)}$, and where $c_i = \cos(2\pi i/n)$.

3 Piecewise smooth subdivision surfaces

Fitting smooth surfaces to non-smooth objects often produces unacceptable results. As an example, fitting an everywhere smooth subdivision surface to the points in Color Plate 1j produces the surface shown in Color Plate 1m.

In this section we develop new subdivision rules to accurately model objects with tangent discontinuities. These subdivision rules produce commonly occurring sharp features that we call *creases*, *corners*, and *darts*, as illustrated in Color Plate 1e–h. A crease is a tangent line smooth curve along which the surface is C^0 but not C^1; a corner is a point where three or more creases meet; finally, a dart is an interior point of a surface where a crease terminates. Although this list of sharp features is not exhaustive (for instance, we cannot model a cone or two coincident darts), it has proven sufficient for the examples we have encountered.

Subdivision surfaces produced by the new rules are tangent plane smooth everywhere except along creases and at corners. A detailed theoretical analysis of the behavior along creases and at corners is beyond the scope of this paper and will be presented in subsequent work. In Section 3.2 we summarize the relevant results of the analysis.

3.1 Subdivision rules: To model creases, corners, and darts using subdivision surfaces, a subset L of edges in the simplicial complex K is tagged as *sharp*. We refer to the pair (K, L) as a *tagged simplicial complex*. The subdivision masks are modified so that tangent plane continuity across sharp edges is relaxed. Boundary curves are produced by tagging all boundary edges of the mesh as sharp.[2] In the subdivision process, edges created through refinement of a sharp edge are tagged as sharp.

Subdivision rules at crease vertices must be chosen carefully in order for the surface on each side of the crease to have a well-defined tangent plane at each point along the crease. Similar considerations apply to corners and darts. It should be noted that the specific subdivision masks we use are by no means unique. Indeed, there is considerable flexibility in selecting them. The masks we present here are simple and have worked well in practice, but further research should be done to explore other alternatives.

We classify vertices into five different types based on the number and arrangement of incident edges. A *smooth vertex* is one where the number of incident sharp edges s is zero; a *dart vertex* has $s = 1$; a *crease vertex* has $s = 2$; and a *corner vertex* has $s > 2$. Crease vertices are further classified as *regular* and *non-regular* depending on the arrangement of smooth edges. An interior crease vertex is regular if it has valence 6 with exactly two smooth edges on each side of the crease; a boundary crease vertex is regular if it has valence 4. All other crease vertices, whether interior or boundary, are non-regular.

Figure 3 shows our vertex and edge subdivision masks. As indicated in the figure, vertex subdivision masks are chosen based on the type of the vertex.

[2] In related work, Nasri [15, 16] developed a method to model boundary curves in a Doo-Sabin subdivision procedure by augmenting the control mesh rather than by modifying the subdivision masks.

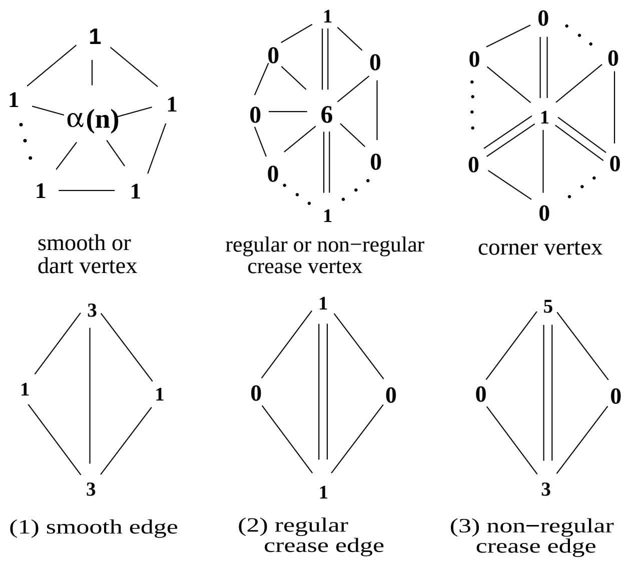

Figure 3: Vertex and edge subdivision masks. Double lines denote sharp edges.

	dart	reg crease	non-reg crease	corner
dart	1	1	1	1
reg crease	1	2	3	3
non-reg crease	1	3	2	2
corner	1	3	2	2

Table 1: Assignment of sharp edge subdivision masks as a function of the types of the two incident vertices. Masks are numbered as shown in Figure 3.

We use three different types of edge subdivision masks. A smooth edge (one not tagged as sharp) is subdivided using the smooth edge mask. The mask used to subdivide a sharp edge depends on the types of the incident vertices as shown in Table 1. When applying edge subdivision mask 3, the regular crease vertex incident to the edge receives the weight 5.

Those familiar with B-spline curve subdivision may recognize that the crease subdivision masks have been designed so that the sharp edges converge to uniform cubic B-splines except near non-regular crease and corner vertices. The zeros in the crease subdivision masks completely decouple the behavior of the surface on one side of the crease from the behavior on the other side.

3.2 Computing surface points and tangent vectors: As explained in Section 2.2, limiting points and tangent planes can be computed using masks. These masks are determined by the eigenstructure of local subdivision matrices, which depend on the type of the vertex (smooth, dart, regular and non-regular crease, and corner).

Smooth and dart vertices: At smooth and dart vertices, our local subdivision matrix is identical to Loop's matrix. The position and tangent masks are therefore as in Figure 2.

Crease vertices: Since the zeros in the crease subdivision masks (Figure 3) decouple the behavior of the surface on one side of the crease from the behavior on the other side, we can decouple the analysis, focusing on a local subdivision matrix that describes the behavior on one side of the crease. As indicated earlier, boundary curves are modeled as one-sided creases.

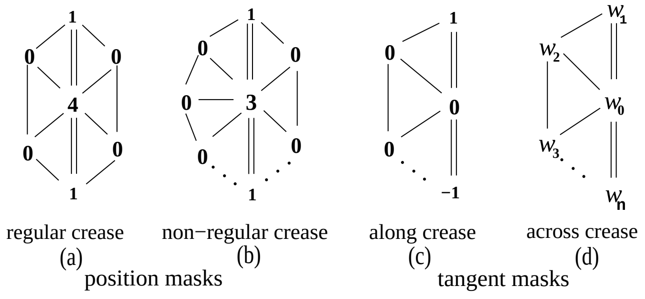

Figure 4: Position and tangent masks for crease vertices.

In the following, we assume that the vertices $\mathbf{v}_1^0, ..., \mathbf{v}_n^0$ surrounding one side of a crease vertex $\mathbf{v}^0$ of valence n are indexed as shown in Figure 4d.

At a regular crease vertex, the dominant left eigenvector of the local subdivision matrix yields the position mask shown in Figure 4a, meaning that

$$\mathbf{v}^{\infty} = \frac{1}{6}(4\mathbf{v}^0 + \mathbf{v}_1^0 + \mathbf{v}_n^0)$$

is a point on the limit crease. Similarly, when the crease vertex is non-regular, we obtain the position mask shown in Figure 4b.

For crease vertices of valence 4 or higher, the subdivision rules described in the previous section give rise to well-defined tangent planes on both sides of the crease.[3] As for smooth vertices, tangent masks are again determined by the two left eigenvectors corresponding to the 2nd and 3rd largest eigenvalues. For both regular and non-regular crease vertices, a tangent along the crease is obtained by the tangent mask shown in Figure 4c. To compute a tangent vector transverse to the crease, we use the tangent mask shown in Figure 4d, where the weights are defined as follows. At a regular crease vertex, the valence is 4 and the mask is given by $(w_0, ..., w_4) = (-2, -1, 2, 2, -1)$. At a non-regular crease vertex, for $n \geq 4$, $w_0 = 0$, $w_1 = w_n = \sin\theta$, and $w_i = (2\cos\theta - 2)(\sin(i-1)\theta)$ for $i = 2, \ldots, (n-1)$ where $\theta = \pi/(n-1)$; for $n = 3$, $(w_0, ..., w_3) = (-1, 0, 1, 0)$; finally, for $n = 2$, $(w_0, w_1, w_2) = (-2, 1, 1)$.

Corner vertices: The subdivision masks at a corner vertex are much like those at a crease vertex. If the corner vertex has s sharp edges, the local subdivision matrix decouples into s separate matrices (or $s-1$ matrices if the corner vertex lies on a boundary), each describing a smooth region of the surface. Since the corner vertex does not move during subdivision, it is itself a point on the surface (equivalently, $(1, 0, ..., 0)$ is the dominant left eigenvector). The tangent masks in this case reduce to simple differences: $(1, -1, 0, ..., 0)$ and $(1, 0, 0, ..., -1)$.

4 Fitting piecewise smooth subdivision surfaces

In this section, we describe an algorithm for phase 3 of the reconstruction problem as outlined in Section 1.

The input to phase 3 is an unstructured collection $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of data points scattered in three dimensions, together with the mesh obtained from phase 2. Edges are initially tagged as sharp if the dihedral angle of the faces incident to the edge is above a threshold (e.g. 40 degrees). The output of phase 3 is a concise piecewise smooth surface that accurately fits the data.

As in phase 2 (mesh optimization), we cast the problem as one of minimizing an energy function that captures the competing goals of conciseness and accuracy.

[3]The techniques we use to prove smoothness do not apply to vertices of valence 2 and 3, although numerical experiments suggest that tangent planes are well-defined in these cases, too.

4.1 Definition of the energy function: The energy function is given by

$$E(K, L, V) = E_{dist}(K, L, V) + c_{rep}m + c_{sharp}e$$

where E_{dist} is the total squared distance from the data points to the subdivision surface; $c_{rep}m$ is a penalty on the number m of vertices; and $c_{sharp}e$ is a penalty on the number e of sharp edges.

The parameter c_{rep} controls the trade-off between conciseness and fidelity to the data and should be set by the user. The parameter c_{sharp} controls the tradeoff between smoothness of the surface and fidelity to the data. Setting $c_{sharp} = c_{rep}/5$ has worked well in all our examples.

We minimize the energy function over the space $\mathcal{M}$ of *tagged meshes* $M = (K, L, V)$ where K is of the same topological type as the phase 2 mesh, and L is the subset of sharp edges of K. The goal is to find the tagged mesh in $\mathcal{M}$ that minimizes E.

The reader familiar with Hoppe *et al.* [11] will notice the absence of a "spring energy" term, which was introduced to guide the mesh optimization algorithm into a good local energy well. For the type of examples shown in the Color Plates, that energy term has been unnecessary in phase 3.

4.2 Minimization: Our algorithm for energy minimization closely parallels the one presented in Hoppe *et al.* [11]. We decompose the problem into two nested subproblems: an inner, continuous optimization over the control vertex positions V for fixed (K, L), and an outer, discrete optimization over (K, L).

4.2.1 Optimization over V for fixed (K, L): We want to determine

$$E(K, L) = \min_V E_{dist}(K, L, V) + c_{rep}m + c_{sharp}e \ ,$$

the minimum energy for fixed (K, L). Since m and e are fixed, this is equivalent to minimizing the distance energy over the vertex positions V. In the following, V is treated as an $m \times 3$ matrix whose rows contain the (x, y, z) coordinates of the vertices.

Computing the distance energy E_{dist} involves projecting the data points $\mathbf{x}_i$ onto the subdivision surface S. This is not feasible in practice as the surface is defined only as the limit of an infinite process. Instead, we project onto a piecewise linear approximation $\tilde{S}$ to S obtained by subdividing the original mesh r times to produce a refined mesh M^r, then pushing all the vertices of M^r to their limit positions using the position masks. (Typically we use $r = 2$.) Since each of the vertices of M^r can be written as an affine combination of the vertices V of M (using the subdivision rules), and since the limit position of any vertex can be obtained using the position masks, each of the vertices of $\tilde{S}$ can be written as an affine combination of the vertices V. That is, each vertex $\tilde{\mathbf{v}}$ of $\tilde{S}$ can be written as $\tilde{\mathbf{v}} = \mathbf{y}V$, where the entries of the row vector $\mathbf{y}$ can be computed by combining the effects of r-fold subdivision followed by application of a position mask. Moreover, since $\tilde{S}$ is piecewise linear, every point on $\tilde{S}$ — not just the vertices — can be written as an affine combination of the vertices V.

For each data point $\mathbf{x}_i$, let $\mathbf{w}_i$ be the closest point on $\tilde{S}$. As argued above, $\mathbf{w}_i$ can be written as $\mathbf{y}_i V$, meaning that E_{dist} can be expressed as

$$E_{dist} = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{y}_i\, V\|^2 \ .$$

This expression for E_{dist} is quadratic in V. Hence, for fixed $\mathbf{y}_i$, optimizing over V is a linear least-squares problem. Moreover, the vectors $\mathbf{y}_i$ are sparse since the subdivision rules are local.

This suggests an iterative minimization scheme alternating between the following steps:

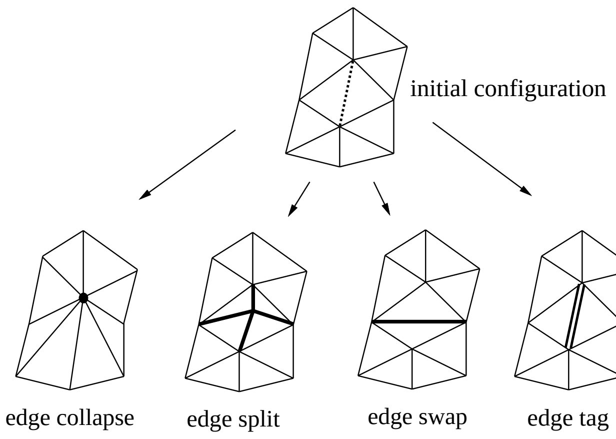

Figure 5: Elementary mesh transformations.

1. For fixed V, compute the projections $\mathbf{y}_i V$ of the data points $\mathbf{x}_i$ onto $\tilde{S}$.

2. For fixed $\mathbf{y}_1, \cdots, \mathbf{y}_n$, optimize E_{dist} over V.

Step 2, which is a sparse linear least squares problem, can be solved as described in [11].

4.2.2 Optimization over (K, L):

Our algorithm for solving the outer minimization problem, minimizing $E(K, L)$, again closely parallels the algorithm of [11].

We define a set of four elementary mesh transformations, *edge collapse*, *edge swap*, *edge split*, and *edge tag*, taking a tagged simplicial complex (K, L) to another tagged simplicial complex (K', L'), as shown in Figure 5. The first three transformations are discussed in [11]. The fourth transformation, edge tag, is a toggle that either adds an edge to the set L of sharp edges, or removes one from it. These four transformations are complete in the sense that they form a transitive set of transformations on the set of tagged simplicial complexes (of a given topological type).

A *legal move* is the application of one of these elementary transformations to an edge of K that leaves the topological type of K unchanged. The criterion for determining whether a move is legal is given in [11]. Our goal is to find a sequence of legal moves taking us from an initial tagged simplicial complex (K_0, L_0) to one for which a minimum of E is achieved.

This is accomplished via a variant of random descent: We form a candidate set, initially consisting of all edges of K. We randomly select an edge from the candidate set and try the four elementary transformations in turn until we find a legal move $(K, L) \Rightarrow (K', L')$ with $E(K', L') < E(K, L)$. If none is found, we remove the edge from the candidate set; otherwise, we accept the move and expand the candidate set to include edges whose vertices were affected by the transformation. The process is repeated until the candidate set is empty.

Due to the expense of computing $E(K', L')$ for each speculative move, the idealized algorithm just described is too inefficient to be of practical use. We therefore replace the exact computation of $E(K', L')$ by an approximate one.

Our approximate computation of E is based on the observation that the effect of an elementary transformation on the geometry of the subdivision surface is localized to a neighborhood of the affected edge. Thus, when speculating upon an elementary transformation, we only optimize over the positions of control vertices in a neighborhood of the affected edge, and recompute projections of data points originally projecting onto the neighborhood of $\tilde{S}$ supported by these control vertices. For details, see [9, 12].

5 Results

The main motivation for moving from piecewise linear to piecewise smooth surfaces is to obtain more accurate and concise models of point sets. The last row of Color Plate 1 and the first 2 rows of Color Plate 2 show results of our experiments with range data. The leftmost column shows the original point sets; the second column shows the optimized piecewise linear surfaces obtained from phase 2 (mesh optimization); the third column shows the optimized tagged meshes resulting from phase 3 (subdivision surface optimization), with sharp edges L highlighted in yellow; finally, the rightmost column shows the subdivision surfaces associated with these control meshes. Modeling surfaces such as the one shown in Color Plate 1t using NURBS would be very cumbersome and would likely require significant user intervention. In contrast, our subdivision surface approach is both simple and automatic.

Our method can also be used for the approximation of known surfaces. To this end, we first generate a set of points on the surface to be approximated, and then run the three phases of the reconstruction procedure. The lower three rows of Color Plate 2 show the resulting approximations of a dense triangular mesh, a swept surface, and a NURBS surface. Since the NURBS teapot was defined as a set of mutually intersecting patches, we had to manually remove some of the sample points. Note how this teapot is modeled as a single subdivision surface of genus 1 (the handle of the teapot makes it homeomorphic to a torus), without resort to explicit continuity constraints or trimming curves.

Another advantage of optimizing using a piecewise smooth model is that the resulting surface not only fits the data more accurately than a piecewise linear model, but also is a better predictor of the true underlying surface. As a validation, we sampled a different set of 10,000 points from the swept surface (knot). As shown in Table 2, even though the subdivision control mesh has a fifth as many vertices as the mesh from phase 2, the subdivision surface fits the new set of points with one fourth the distance energy.

	c_{rep}	m	E_{dist}	
		# vertices	original points	new points
phase 2	10^{-5}	975	.00308	.00934
phase 3	10^{-5}	363	.00042	.00054
	10^{-4}	207	.00216	.00251

Table 2: Validation results

Color	n	c_{rep}		m (#vertices)			E_{dist}			Time
Plate		ph2	ph3	ph2	ph3	$\frac{ph2}{ph3}$	ph2	ph3	$\frac{ph2}{ph3}$	hrs
1j	4,102	10^{-5}	10^{-5}	163	112	1.5	$4.86\ {}_{10}{}^{-4}$	$1.53\ {}_{10}{}^{-4}$	3.2	0.9
1q	30,937	10^{-5}	10^{-5}	891	656	1.4	$4.93\ {}_{10}{}^{-3}$	$4.14\ {}_{10}{}^{-3}$	1.2	7.7
2.1	16,864	10^{-5}	10^{-5}	262	156	1.7	$2.19\ {}_{10}{}^{-3}$	$1.33\ {}_{10}{}^{-3}$	1.6	2.8
2.2	12,745	10^{-5}	10^{-5}	685	507	1.4	$4.05\ {}_{10}{}^{-3}$	$3.85\ {}_{10}{}^{-3}$	1.1	4.1
2.3	16,475	10^{-5}	10^{-4}	184	87	2.1	$9.68\ {}_{10}{}^{-4}$	$1.65\ {}_{10}{}^{-3}$	0.6	2.2
2.4	10,000	10^{-5}	10^{-4}	975	205	4.8	$3.08\ {}_{10}{}^{-3}$	$2.32\ {}_{10}{}^{-3}$	1.3	2.2
2.5	26,103	10^{-5}	10^{-4}	623	152	4.1	$3.17\ {}_{10}{}^{-3}$	$2.62\ {}_{10}{}^{-3}$	1.2	5.3

Table 3: Parameter settings and optimization results

In most examples, the representation constant c_{rep} was set to 10^{-5}, the same value that was used in phase 2. As indicated in Table 3, the control meshes obtained from phase 3 are more concise than those of phase 2, and at the same time, the subdivision surfaces fit the points more accurately than the triangular meshes of phase 2. Because the point sets for the swept surface and the NURBS teapot are sampled without error from piecewise smooth surfaces, we could afford to raise c_{rep} in order to produce very concise control meshes, while still reducing E_{dist}.

The phase 3 execution times listed in Table 3 were obtained on a DEC Alpha workstation. In all test cases we set $c_{sharp} = c_{rep}/5$ and the number of subdivision iterations (referred to in Section 4.2.1) to $r = 2$.

6 Related work

There is a large body of literature on reconstructing surfaces of fixed topological type. Bolle and Vemuri [1] review methods for fitting embeddings of a rectangular domain. Schudy and Ballard [22, 23], Brinkley [2], and Sclaroff and Pentland [24] fit embeddings of a sphere. Schmitt *et al.* [20, 21] fit embeddings of a cylinder to data from cylindrical range scans. Goshtasby [7] works with embeddings of cylinders and tori.

There is also extensive literature on smooth interpolation of triangulated data of arbitrary topological type using parametric surface patches; see Lounsbery *et al.* [14] for a survey. These schemes are designed to interpolate sparse data, rather than to fit dense, noisy point sets of the type obtained from range scanners.

Two recent articles describing methods for fitting either piecewise linear or everywhere smooth surfaces of arbitrary topological type are Veltkamp [27] and Szeliski *et al.* [26].

We are not aware of any previous method for fitting piecewise smooth surface models of arbitrary topological type to dense, noisy data, although one could imagine developing such a procedure based on a piecewise smooth triangular patch method such as Nielson's side-vertex patch [17], or Shirman and Séquin's split domain scheme [25].

In many respects, our work can be considered a generalization to surfaces of the parametric curve fitting method of Plass and Stone [18]: they cast the fitting process as non-linear optimization, and they also produce piecewise smooth, rather than everywhere smooth models.

7 Summary and future work

We have described a piecewise smooth surface reconstruction procedure that produces concise and accurate surface models from unorganized points. Our method automatically determines the topological type of the surface, and the presence and location of sharp features. A key ingredient of the method is a new subdivision surface scheme that allows the modeling of surface features such as corners, boundaries, creases, and darts. Finally, we have demonstrated the effectiveness of the subdivision surface optimization procedure in recovering piecewise smooth models from range data, and in approximating other surface forms such as swept surfaces and NURBS.

There are a number of areas for future research, including:

1. Development of subdivision rules that can model a wider variety of sharp features such as cones, multiple darts meeting at a smooth vertex, and darts meeting at a corner.
2. Development of alternative optimization algorithms that allow direct control over maximum error.
3. Speedup of the algorithm and implementations on parallel architectures.
4. Development of an on-line algorithm for use in real-time data capture.
5. Experimentation with sparse, non-uniform data.

Acknowledgments We wish to thank the Ford Motor Co. for providing the cross section data, Technical Arts Co. for the laser range data, Rob Sharein for the swept surface, and Pratt & Whitney for the dense tessellation of the gas turbine engine component.

References

[1] Ruud M. Bolle and Baba C. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Trans. Pat. Anal. Mach. Intell.*, 13(1):1–13, January 1991.

[2] James F. Brinkley. Knowledge-driven ultrasonic three-dimensional organ modeling. *IEEE Trans. Pat. Anal. Mach. Intell.*, 7(4):431–441, July 1985.

[3] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, September 1978.

[4] T. DeRose, H. Hoppe, T. Duchamp, J. McDonald, and W. Stuetzle. Fitting of surfaces to scattered data. *SPIE*, 1830:212–220, 1992.

[5] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, September 1978.

[6] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 3rd edition, 1992.

[7] Ardeshir Goshtasby. Surface reconstruction from scattered measurements. *SPIE*, 1830:247–256, 1992.

[8] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 35–44, August 1993.

[9] H. Hoppe, T. DeRose, T. Duchamp, H. Jin, J. McDonald, and W. Stuetzle. Piecewise smooth surface reconstruction. TR 94-01-01, Dept. of Computer Science and Engineering, University of Washington, January 1994.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):71–78, July 1992.

[11] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 19–26, August 1993.

[12] Hugues Hoppe. *Surface reconstruction from unorganized points*. PhD thesis, Department of Computer Science and Engineering, University of Washington, In preparation.

[13] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.

[14] Michael Lounsbery, Stephen Mann, and Tony DeRose. Parametric surface interpolation. *IEEE Computer Graphics and Applications*, 12(5):45–52, September 1992.

[15] Ahmad H. Nasri. Polyhedral subdivision methods for free-form surfaces. *ACM Transactions on Graphics*, 6(1):29–73, January 1987.

[16] Ahmad H. Nasri. Boundary-corner control in recursive-subdivision surfaces. *Computer Aided Design*, 23(6):405–410, July-August 1991.

[17] G. Nielson. A transfinite, visually continuous, triangular interpolant. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 235–246. SIAM, 1987.

[18] Michael Plass and Maureen Stone. Curve-fitting with piecewise parametric cubics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):229–239, July 1983.

[19] Ulrich Reif. A unified approach to subdivision algorithms. Mathematisches Institute A 92-16, Universität Stuttgart, 1992.

[20] F. Schmitt, B.A. Barsky, and W. Du. An adaptive subdivision method for surface fitting from sampled data. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4):179–188, 1986.

[21] F. Schmitt, X. Chen, W. Du, and F. Sair. Adaptive G^1 approximation of range data using triangular patches. In P.J. Laurent, A. Le Mehaute, and L.L. Schumaker, editors, *Curves and Surfaces*. Academic Press, 1991.

[22] R. B. Schudy and D. H. Ballard. Model detection of cardiac chambers in ultrasound images. Technical Report 12, Computer Science Department, University of Rochester, 1978.

[23] R. B. Schudy and D. H. Ballard. Towards an anatomical model of heart motion as seen in 4-d cardiac ultrasound data. In *Proceedings of the 6th Conference on Computer Applications in Radiology and Computer-Aided Analysis of Radiological Images*, 1979.

[24] S. Sclaroff and A. Pentland. Generalized implicit functions for computer graphics. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):247–250, July 1991.

[25] L. Shirman and C. Séquin. Local surface interpolation with Bézier patches. *Computer Aided Geometric Design*, 4(4):279–296, 1988.

[26] R. Szeliski, D. Tonnesen, and D. Terzopoulos. Modeling surfaces of arbitrary topology with dynamic particles. In *1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 82–87. IEEE Computer Society Press, 1993.

[27] R.C. Veltkamp. 3D computational morphology. *Computer Graphics Forum*, 12(3):116–127, 1993.