Generating Smooth Surfaces with Bicubic Splines over Triangular Meshes: Toward Automatic Model Building From Unorganized 3D Points

Ueshiba, T.; Roth, Gerhard

*Generating Smooth Surfaces with Bicubic Splines over Triangular Meshes**

T. Ueshiba and G. Roth
October 1999

***published in** Proceedings of the Second International Conference on Recent Advances in 3-D Digital Imaging and Modeling (3DIM'99),Ottawa, On. October 4-8, 1999. pp. 302-311. NRC 43636.

Generating Smooth Surfaces with Bicubic Splines over Triangular Meshes:

Toward Automatic Model Building from Unorganized 3D Points

Toshio Ueshiba
Intelligent Systems Division
Electrotechnical Laboratory
Tsukuba-shi, Ibaraki, 305–8568 Japan
ueshiba@etl.go.jp

Gerhard Roth
Institute for Information Technology
National Research Council of Canada
Ottawa, K1A0R6 Canada
Gerhard.Roth@iit.nrc.ca

Abstract

This paper presents a new algorithm for constructing tangent plane continuous (G^1) surfaces with piecewise polynomials over triangular meshes. The input mesh can be of arbitrary topological type, that is, any number of faces can meet at a mesh vertex. The mesh is first refined to one solely with quadrilateral cells. Rectangular Bézier patches are then assigned to each of the cells and control points are determined so that G^1 continuity across the patch boundaries is maintained. Since all the patches are rectangular, the resulting surface can be rendered efficiently by current commercial graphic hardware/software. In addition, by exploiting the fact that all the faces of the original mesh are triangular, the degree of each patch is optimized to three while more general method dealing with arbitrary irregular meshes requires biquartic patches. Several surface examples generated from real 3D data are shown.

1 Introduction

Recent progress in range sensing technology has made it possible to directly capture 3D information of an object with great accuracy. This development makes it possible to create CAD models of existing organic or sculptured shapes automatically. Once such models have been acquired, it is possible to incorporate or modify existing parts into a new design, which is referred to as *reverse engineering*.

An important issue in creating a CAD model is the descriptive ability of its representation. Tensor-product non-rational/rational B-spline surfaces[3, 10] are the most widely used representations for modeling free-form objects. There is, however, a severe limitation with these surfaces; they can model only a small subclass of surfaces which are topologically isomorphic to planes or tori. This is because the tensor-product surfaces have a *regular* structure, that is, every patch is quadrilateral (4-sided) and exactly four patches meet at every vertex. In order to model a wider class of objects, therefore, a surface reconstruction method over *irregular* meshes needs to be developed.

Maintaining smoothness between the patches is a key problem in shape description with piecewise polynomials. B-splines can represent complicated free-form surfaces while automatically keeping parametric continuity (C^n). Surfaces defined over irregular meshes, however, cannot attain C^n continuity because no uniform parameterization is possible. This difficulty can be overcome by relaxing the requirement of parametric continuity to that of geometric continuity (G^n). As for first derivatives, G^1 continuity means that the associated surface normal varies smoothly when a point moves over the surface. In this paper, we shall restrict our concern only to G^1 continuity.

In order to make the problem tractable, most of the methods for constructing G^1 surfaces proposed so far refine meshes of arbitrary topology into simpler ones instead of directly handling them in their original forms.

The first approach is to refine general irregular meshes into ones allowing faces with arbitrary number of edges but vertices of 4-valent only. (The term *valence* denotes the number of faces meeting at a vertex.) S-patch[8] yields G^1 surfaces over meshes of this type and its quadratic version has been successfully applied to smooth surface reconstruction from unorganized range data sets[13]. Computing the control points of S-patches, however, is computationally rather expensive for faces with a large number of edges, e.g. over 6 or 7. Moreover, most of the current graphic hardware supports high-speed rendering only for tensor-product non-rational/rational B-spline surfaces. It is therefore desirable that the mesh consists of only quadrilateral faces.

Another approach which exhibits duality with the above meets this requirement; An n-sided polygon can be parti-

0-7695-0062-5/99 $10.00 © 1999 IEEE

tioned into n quadrilaterals by inserting new vertices at the midpoints of each edge and connecting them with the centroid of the n-gon. Applying this midpoint refinement, we can convert general irregular meshes into ones with solely quadrilateral faces. Moreover, this mesh has a distinctive feature that every vertex adjacent to the irregular (non 4-valent) vertex is regular (4-valent) which makes it simple to construct a smooth surface over this mesh. Loop[7] proposed a G^1 surface construction technique with polynomial patches. It requires, however, not only quadrilateral but also triangular patches of up to degree four. Peters[9] presented a simple method of generating smooth surfaces with biquartic (4×4 degree) rectangular Bézier patches. Reif[11] showed it possible to construct a G^1 spline surface with biquadratic patches. In order to apply this technique, however, the refinement process must be repeated until all the irregular vertices are surrounded by three or more layers of regular faces.

In contrast to the above-mentioned techniques dealing with general meshes, we here propose a novel method of constructing G^1 surfaces over triangular meshes. Since the triangular mesh is the most common polygonal representation for objects with general topological type, it is sensible to develop a method specialized for triangular ones. The input to our algorithm is a triangular mesh built from multi-view 3D data[4, 12, 1]. Each triangle in the mesh is then partitioned into three quadrilaterals called *cells*. These cells are represented by tensor-product bicubic Bézier patches. All the control points are determined so that G^1 continuity is maintained between adjacent patches. By exploiting the fact that all the mesh faces are triangular, degree of the patches is constrained to three while the more general method[9] requires patches of degree four. In addition, since all the patches are rectangular, the generated surface can benefit in rendering from current commercial graphic hardware/software in contrast to the G^1 triangular splines [6] composed of Bézier triangles up to degree-6.

This paper is organized as follows: We begin by deriving sufficient conditions for G^1 continuity between two Bézier patches in section 2. The first step of the proposed algorithm is to refine given triangular meshes and compute intermediate points called *generating points*. This is discussed in section 3. Section 4, the key part of this paper, presents a new algorithm to compute control points of bicubic G^1 spline surfaces defined over the refined meshes. We validate the proposed method through experiments in section 5. Finally we give concluding remarks and the future direction of our work.

2 Sufficient conditions for G^1 continuity between Bézier rectangles

In this section, we first derive sufficient conditions for G^1 continuity between two Bézier patches. Then we reinterpret these conditions in terms of constraints at mesh vertices.

2.1 Continuity between two Bézier patches

We here consider two rectangular $d \times d$ degree Bézier patches defined by

$$\begin{aligned} S(u,v) &= \sum_{i=0}^{d}\sum_{j=0}^{d} B_i^d(u)B_j^d(v)\mathbf{P}_{ij} \\ S'(u,v) &= \sum_{i=0}^{d}\sum_{j=0}^{d} B_i^d(u)B_j^d(v)\mathbf{P}'_{ij} \end{aligned} \qquad (u,v \in [0,1])$$

where $\mathbf{P}_{ij}$ and $\mathbf{P}'_{ij}$ are control points in 3-space and

$$B_i^d(u) \equiv \binom{d}{i} u^i(1-u)^{d-i} \qquad (i = 0,\dots,d)$$

are Bernstein polynomials. Suppose that these two patches meet at common boundary B which forms a degree-d Bézier curve $S(u,0) = S'(u,0)$. This implies S and S' share common control points along B, that is, $\mathbf{P}_{i0} = \mathbf{P}'_{i0}$ $(i = 0,\dots,d)$. Versal and transversal derivatives of S and S' along B are then computed by[3]

$$\begin{aligned} \frac{\partial S}{\partial u}(u,0) = \frac{\partial S'}{\partial u}(u,0) &= d\sum_{i=0}^{d-1} B_i^{d-1}(u)(\mathbf{P}_{i+1\,0} - \mathbf{P}_{i0}) \\ \frac{\partial S}{\partial v}(u,0) &= d\sum_{i=0}^{d} B_i^d(u)(\mathbf{P}_{i1} - \mathbf{P}_{i0}) \\ \frac{\partial S'}{\partial v}(u,0) &= d\sum_{i=0}^{d} B_i^d(u)(\mathbf{P}'_{i1} - \mathbf{P}_{i0}). \end{aligned}$$

Since a tangent plane of S (resp. S') at a point on B is spanned by $\frac{\partial S}{\partial u}$ and $\frac{\partial S}{\partial v}$ (resp. $\frac{\partial S'}{\partial u}$ and $\frac{\partial S'}{\partial v}$), tangent plane ($G^1$) continuity between S and S' implies co-planarity of these three vectors, i.e. $\phi(u)\frac{\partial S}{\partial v}(u,0) + \phi'(u)\frac{\partial S'}{\partial v}(u,0) = \psi(u)\frac{\partial S}{\partial u}(u,0)$ holds for some scalar-value functions ϕ, ϕ' and ψ. We here consider a special case in which

$$\begin{aligned} &\frac{1}{2}\sum_{i=0}^{d} B_i^d(u)(\mathbf{P}_{i1} - \mathbf{P}_{i0}) + \frac{1}{2}\sum_{i=0}^{d} B_i^d(u)(\mathbf{P}'_{i1} - \mathbf{P}_{i0}) \\ &= \{\lambda(1-u) - \mu u\}\sum_{i=0}^{d-1} B_i^{d-1}(u)(\mathbf{P}_{i+1\,0} - \mathbf{P}_{i0}) \end{aligned}$$

holds for unknown scalars λ and μ. Noting that the right side of the equation above can be rewritten as

$$\{\lambda(1-u) - \mu u\}\sum_{i=0}^{d-1} B_i^{d-1}(u)(\mathbf{P}_{i+1\,0} - \mathbf{P}_{i0})$$

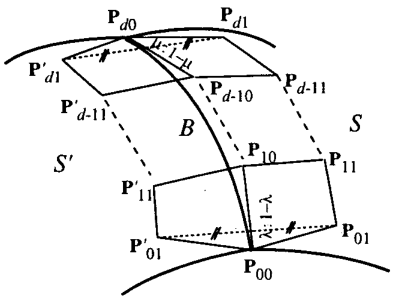


Figure 1. Continuity between two Bézier patches.

$$= \sum_{i=0}^{d-1}\{\lambda(1-u)B_i^{d-1}(u) - \mu u B_i^{d-1}(u)\}(\mathbf{P}_{i+1\,0} - \mathbf{P}_{i0})$$
$$= \sum_{i=0}^{d-1}\left\{\lambda\frac{d-i}{d}B_i^d(u) - \mu\frac{i+1}{d}B_{i+1}^d(u)\right\}(\mathbf{P}_{i+1\,0}-\mathbf{P}_{i0})$$
$$= \sum_{i=0}^{d} B_i^d(u)\left\{\left(1-\frac{i}{d}\right)\lambda(\mathbf{P}_{i+1\,0} - \mathbf{P}_{i0}) - \frac{i}{d}\mu(\mathbf{P}_{i0} - \mathbf{P}_{i-1\,0})\right\}$$

we have

$$\frac{\mathbf{P}_{i1} + \mathbf{P}'_{i1}}{2} = \frac{i}{d}\mu\mathbf{P}_{i-1\,0} + \left\{1 - \frac{i}{d}\mu - \left(1-\frac{i}{d}\right)\lambda\right\}\mathbf{P}_{i0} + \left(1-\frac{i}{d}\right)\lambda\mathbf{P}_{i+1\,0} \quad (i=0,\ldots,d) \quad (1)$$

by equating and rearranging the coefficients of $B_i^d(u)$. Equation (1) yields sufficient conditions for G^1 continuity between the two patches S and S'.

2.2 Tangent and twist constraints

Now we rewrite (1) for $i = 0$ as

$$\frac{\mathbf{P}_{01} + \mathbf{P}'_{01}}{2} = (1-\lambda)\mathbf{P}_{00} + \lambda\mathbf{P}_{10} \quad (2)$$

which implies that the midpoint of $\overline{\mathbf{P}_{01}\mathbf{P}'_{01}}$ is on $\overline{\mathbf{P}_{00}\mathbf{P}_{10}}$ and its separation from $\mathbf{P}_{00}$ is represented by λ (see Fig. 1). We therefore call λ *displacement factor*. In addition, the plane defined by $\mathbf{P}_{00}$, $\mathbf{P}_{10}$, $\mathbf{P}_{01}$ and $\mathbf{P}'_{01}$ is a tangent plane of the resulting G^1 surface at $\mathbf{P}_{00}$. We therefore call (2) *tangent constraint.*

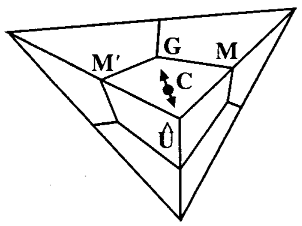


Figure 2. Triangles partitioned into three quadrilaterals.

Similarly, rewriting (1) for $i = d$ gives the tangent constraint at the other end $\mathbf{P}_{d0}$ of the boundary and μ is a displacement factor at $\mathbf{P}_{d0}$.

We also rewrite (1) for $i = 1$ as

$$\frac{\mathbf{P}_{11} + \mathbf{P}'_{11}}{2} = \frac{\mu}{d}\mathbf{P}_{00} + \left\{1 - \frac{\mu}{d} - \left(1-\frac{1}{d}\right)\lambda\right\}\mathbf{P}_{10} + \left(1-\frac{1}{d}\right)\lambda\mathbf{P}_{20}. \quad (3)$$

Since $\mathbf{P}_{11}$ (resp. $\mathbf{P}'_{11}$) determines the twist, that is, cross derivative with respect to two parameters u and v, of S (resp. S') at $\mathbf{P}_{00}$, we call (3) *twist constraint.*

Similarly, the twist constraint at $\mathbf{P}_{d0}$ is given by rewriting (1) for $i = d-1$.

3 Mesh refinement and computation of generating points

Suppose that a closed triangular mesh with arbitrary topology is given. The first step of the proposed surface construction algorithm is to refine this mesh into one solely with quadrilateral faces called *cells*. This is accomplished by inserting a new point (type G) at the centroid of each triangular face and connecting it with three points (type M) inserted at the middle of the surrounding edges. Though valence, i.e. the number of faces meeting at a vertex, of vertices of the original mesh can be an arbitrary number greater than two, the valence of type G and type M vertices are always three and four respectively.

Next, we compute the intermediate points called *generating points* and smooth the mesh using them. We here adopt a simplified version of Peters' blending method[9].

Let $\hat{\mathbf{U}}$ be a vertex of the original mesh and $\mathbf{M}$ and $\mathbf{M}'$ denote midpoints of two edges incident on $\hat{\mathbf{U}}$. In addition, let $\mathbf{G}$ represent the centroid of the triangle involving these edges (see Fig. 2). Then the generating point for the quadrilateral $\hat{\mathbf{U}}\mathbf{M}\mathbf{G}\mathbf{M}'$ is computed as a convex combination of

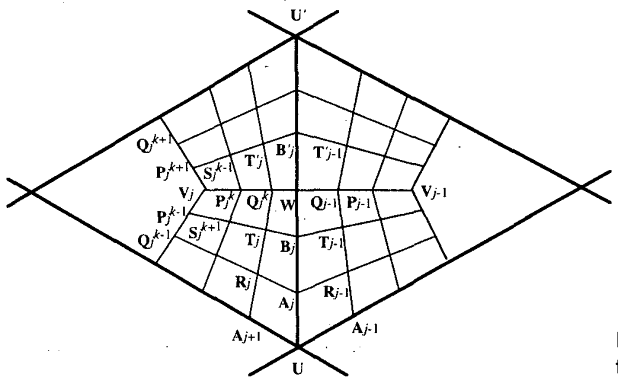


Figure 3. Labeling scheme for Bézier control points.

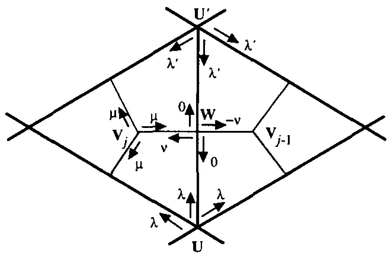


Figure 4. Configuration of displacement factors.

these four points such that

$$\begin{aligned} \mathbf{C} \;\equiv\; & (1-a)^2\hat{\mathbf{U}} + (1-a)a\mathbf{M} \\ & +(1-a)a\mathbf{M}' + a^2\mathbf{G} \qquad (0 < a < 1). \quad (4) \end{aligned}$$

The parameter a is called *blend ratio* and controls the shape of the final surface. If a increases, $\mathbf{C}$ is pulled toward $\mathbf{G}$ and a surface with round shape results. Conversely, $\mathbf{C}$ approaches $\hat{\mathbf{U}}$ for small value of a and produces a flat shaped surface close to the original triangular mesh.

We then smooth the mesh by shifting $\hat{\mathbf{U}}$ toward the centroid of n generating points $\mathbf{C}_j$ $(j = 0, \dots, n{-}1)$ surrounding it:

$$\mathbf{U} \equiv (1-\alpha)\hat{\mathbf{U}} + \frac{\alpha}{n}\sum_{j=0}^{n-1}\mathbf{C}_j \qquad (0 < \alpha < 1) \quad (5)$$

where n stands for valence of $\hat{\mathbf{U}}$. The shift parameter α also plays a role of shape controller as well as a.

4 Bicubic G^1 spline surfaces over refined meshes

In this section, we construct a G^1 continuous surface over the refined mesh obtained in the previous section. We assign a bicubic Bézier patch to each quadrilateral cell of the mesh and determine control points so that the adjacent patches join in a tangent plane continuous manner.

We use the labeling scheme shown in Fig. 3 for control points of the Bézier patches. $\mathbf{U}$ defined by (5) stands for an n-valent vertex of the refined and smoothed mesh. We use subscript j to index control points surrounding $\mathbf{U}$. $\mathbf{V}_j$ represents a 3-valent vertex corresponding to the centroid of the original triangular face. We also introduce superscript k to index control points around $\mathbf{V}_j$. Indices j and k must be taken as modulo n and 3 respectively. $\mathbf{W}$ is a 4-valent vertex which corresponds to the midpoint of the original edge.

Since all the patches are bicubic ($d = 3$), we have four constraints derived in §2.2 for each boundary between two patches: two tangent constraints and two twist constraints at both ends. Consequently, satisfying tangent and twist constraints at every vertex of the mesh is enough to guarantee overall G^1 continuity.

4.1 Displacement factors

Displacement factors introduced in §2.2 must be set to appropriate values to attain G^1 continuity of the surface. Figure 4 shows their configuration used in the proposed algorithm. For $\mathbf{U}$ and $\mathbf{V}_j$, we set displacement factors to the same value λ and μ respectively for every direction. For $\mathbf{W}$, however, four displacement factors are chosen to be different values: zero for boundaries toward $\mathbf{U}$ and $\mathbf{U}'$, ν toward $\mathbf{V}_j$ and $-\nu$ toward $\mathbf{V}_{j-1}$. If ν is non-zero, two patch boundaries toward $\mathbf{U}$ and $\mathbf{U}'$ form a bend at $\mathbf{W}$. The parameter ν introduces an additional degree of freedom which is essential to satisfy twist constraints at $\mathbf{W}$. This will be clear in §4.2.

4.2 Construction of the control points

Now the control points are determined so that tangent and twist constraints are satisfied at every vertex.

First, we consider constraints at n-valent vertex $\mathbf{U}$. From the configuration of the displacement factors shown in Fig. 4, tangent and twist constraints ((2) and (3) respectively) at

$\mathbf{U}$ have the following forms:

$$\begin{aligned}&\frac{\mathbf{A}_{j-1}+\mathbf{A}_{j+1}}{2}\\&= (1-\lambda)\mathbf{U}+\lambda\mathbf{A}_j \qquad (j=0,\dots,n-1), \qquad (6)\\&\frac{\mathbf{R}_{j-1}+\mathbf{R}_j}{2}\\&= \left(1-\frac{2}{3}\lambda\right)\mathbf{A}_j+\frac{2}{3}\lambda\mathbf{B}_j \quad (j=0,\dots,n-1). \quad (7)\end{aligned}$$

Following Peters' scheme[9], we define new intermediate points $\mathbf{D}_j$ $(j=0,\dots,n-1)$ from the generating points $\mathbf{C}_0,\dots,\mathbf{C}_{n-1}$ around $\mathbf{U}$ as

$$\mathbf{D}_j \equiv \mathbf{U}+\frac{\alpha}{n}\sum_{k=0}^{n-1}\cos\frac{2\pi(j-k)}{n}\mathbf{C}_k$$
$$(0<\alpha<1;\ j=0,\dots,n-1). \qquad (8)$$

Since $\mathbf{D}_j$ satisfies the following relation (see appendix A)

$$\frac{\mathbf{D}_{j-1}+\mathbf{D}_{j+1}}{2}=\left(1-\cos\frac{2\pi}{n}\right)\mathbf{U}+\cos\frac{2\pi}{n}\mathbf{D}_j, \qquad (9)$$

tangent constraints (6) are satisfied by defining λ and $\mathbf{A}_j$ as

$$\left\{\begin{aligned}\lambda &\equiv \cos\frac{2\pi}{n}\\ \mathbf{A}_j &\equiv \frac{\mathbf{D}_{j-1}+\mathbf{D}_j}{2} \qquad (j=0,\dots,n-1)\end{aligned}\right. . \qquad (10)$$

The twist constraints (7) are then satisfied by defining $\mathbf{B}_j$ and $\mathbf{R}_j$ as

$$\left\{\begin{aligned}\mathbf{B}_j &\equiv \frac{\mathbf{E}_{j-1}+\mathbf{E}_j}{2}\\ \mathbf{R}_j &\equiv \left(1-\frac{2}{3}\lambda\right)\mathbf{D}_j+\frac{2}{3}\lambda\mathbf{E}_j\end{aligned}\right. \qquad (j=0,\dots,n-1) \qquad (11)$$

respectively where $\mathbf{E}_j$ is another intermediate point. Though $\mathbf{E}_j$ can be determined arbitrarily from the viewpoint of G^1 continuity, we choose

$$\mathbf{E}_j=\frac{1}{2}\mathbf{C}_j+\frac{1}{2}\mathbf{D}_j \qquad (j=0,\dots,n-1) \qquad (12)$$

which produces, in our experience, smooth surfaces with pleasing shape for a wide range of values of the blend ratio.

Next, we rewrite tangent constraints (2) for the internal vertex $\mathbf{V}_j$ as

$$\frac{\mathbf{P}_j^{k-1}+\mathbf{P}_j^{k+1}}{2}=(1-\mu)\mathbf{V}_j+\mu\mathbf{P}_j^k \quad (k=0,1,2). \quad (13)$$

It is easy to confirm that (13) is satisfied by defining $\mathbf{V}_j$ as the centroid of three points $\mathbf{P}_j^0$, $\mathbf{P}_j^1$ and $\mathbf{P}_j^2$ and setting $\mu=-\frac{1}{2}$. We therefore determine μ, $\mathbf{P}_j^k$ and $\mathbf{V}_j$ as

$$\left\{\begin{aligned}\mu &\equiv -\frac{1}{2}\\ \mathbf{P}_j^k &\equiv \frac{\mathbf{E}_j^{k+1}+\mathbf{E}_j^{k-1}}{2} \qquad (k=0,1,2)\\ \mathbf{V}_j &\equiv \frac{\mathbf{E}_j^0+\mathbf{E}_j^1+\mathbf{E}_j^2}{3}\end{aligned}\right. \qquad (14)$$

where $\mathbf{E}_j^k$ $(k=0,1,2)$ are the intermediate points defined by (12) surrounding $\mathbf{V}_j$.

Thirdly, let us consider constraints at $\mathbf{W}$. Here we drop superscript k for notation convenience. From the configuration shown in Fig. 4, the tangent constraints (2) become:

$$\begin{aligned}\frac{\mathbf{Q}_{j-1}+\mathbf{Q}_j}{2} &= \mathbf{W} &(15)\\ \frac{\mathbf{B}_j+\mathbf{B}'_j}{2} &= (1-\nu)\mathbf{W}+\nu\mathbf{Q}_j\\ &= (1+\nu)\mathbf{W}-\nu\mathbf{Q}_{j-1}. &(16)\end{aligned}$$

On the other hand, also from the configuration shown in Fig. 4, the twist constraints (3) in four directions toward $\mathbf{U}$, $\mathbf{V}_{j-1}$, $\mathbf{U}'$ and $\mathbf{V}_j$ are

$$\begin{aligned}\frac{\mathbf{T}_{j-1}+\mathbf{T}_j}{2} &= \frac{\lambda}{3}\mathbf{W}+\left(1-\frac{\lambda}{3}\right)\mathbf{B}_j &(17)\\ \frac{\mathbf{T}_{j-1}+\mathbf{T}'_{j-1}}{2} &= -\frac{1}{6}\mathbf{W}+\left(\frac{7}{6}+\frac{2}{3}\nu\right)\mathbf{Q}_{j-1}-\frac{2}{3}\nu\mathbf{P}_{j-1} &(18)\\ \frac{\mathbf{T}'_{j-1}+\mathbf{T}'_j}{2} &= \frac{\lambda'}{3}\mathbf{W}+\left(1-\frac{\lambda'}{3}\right)\mathbf{B}'_j &(19)\\ \frac{\mathbf{T}_j+\mathbf{T}'_j}{2} &= -\frac{1}{6}\mathbf{W}+\left(\frac{7}{6}-\frac{2}{3}\nu\right)\mathbf{Q}_j+\frac{2}{3}\nu\mathbf{P}_j &(20)\end{aligned}$$

respectively because, from (14), we already know that the displacement factors at $\mathbf{V}_j$ and $\mathbf{V}_{j-1}$ equal $-\frac{1}{2}$. Noting that (17) − (18) + (19) − (20) = 0 holds, we have

$$\left(\frac{1}{3}+\frac{\lambda+\lambda'}{3}\right)\mathbf{W}-\left(\frac{7}{6}+\frac{2}{3}\nu\right)\mathbf{Q}_{j-1}-\left(\frac{7}{6}-\frac{2}{3}\nu\right)\mathbf{Q}_j$$
$$+\left(1-\frac{\lambda}{3}\right)\mathbf{B}_j+\left(1-\frac{\lambda'}{3}\right)\mathbf{B}'_j-\frac{2}{3}\nu(\mathbf{P}_j-\mathbf{P}_{j-1})=0$$

which produces

$$\left(1-\frac{\lambda+\lambda'}{6}+\frac{2}{3}\nu\right)\mathbf{Q}_{j-1}+\left(1-\frac{\lambda+\lambda'}{6}-\frac{2}{3}\nu\right)\mathbf{Q}_j$$
$$=\left(1-\frac{\lambda}{3}\right)\mathbf{B}_j+\left(1-\frac{\lambda'}{3}\right)\mathbf{B}'_j-\frac{2}{3}\nu(\mathbf{P}_j-\mathbf{P}_{j-1}) \quad (21)$$

by using (15) and rearranging. On the other hand, eliminating $\mathbf{W}$ from (15) and (16), we obtain

$$(1-\nu)\mathbf{Q}_{j-1}+(1+\nu)\mathbf{Q}_j=\mathbf{B}_j+\mathbf{B}'_j. \qquad (22)$$

Equations (21) and (22) can be regarded as a linear system with two unknowns $\mathbf{Q}_{j-1}$ and $\mathbf{Q}_j$. If ν is zero, this system has no solutions unless $\lambda = \lambda'$ holds, which means the valence of $\mathbf{U}$ is equal to that of $\mathbf{U}'$. Parameter ν therefore plays an essential role in our surface construction procedure to guarantee the existence of the solution even if the valence of $\mathbf{U}$ is different from that of $\mathbf{U}'$. The solution for $\mathbf{Q}_{j-1}$ and $\mathbf{Q}_j$ is given by

$$\left\{\begin{array}{rcl} \mathbf{Q}_{j-1} &=& \frac{1}{10-\lambda-\lambda'}\left[\left\{(5-\lambda)-\frac{1}{2c}\right\}\mathbf{B}_j \right. \\ && +\left\{(5-\lambda')+\frac{1}{2c}\right\}\mathbf{B}'_j \\ && \left. -\{2+2c(\lambda-\lambda')\}(\mathbf{P}_j-\mathbf{P}_{j-1})\right] \\ \mathbf{Q}_j &=& \frac{1}{10-\lambda-\lambda'}\left[\left\{(5-\lambda)+\frac{1}{2c}\right\}\mathbf{B}_j \right. \\ && +\left\{(5-\lambda')-\frac{1}{2c}\right\}\mathbf{B}'_j \\ && \left. +\{2-2c(\lambda-\lambda')\}(\mathbf{P}_j-\mathbf{P}_{j-1})\right] \end{array}\right. \quad (23)$$

where ν is determined as

$$\nu \equiv c(\lambda - \lambda') \quad (24)$$

for an arbitrary non-zero scalar c (see Appendix B).

Once $\mathbf{Q}_{j-1}$ and $\mathbf{Q}_j$ are computed, we can set four twist points around $\mathbf{W}$ as follows:

$$\left\{\begin{array}{rcl} \mathbf{T}_{j-1} &\equiv& \mathbf{K}_{j-1}+\frac{\mathbf{J}-\mathbf{J}'}{2} \\ \mathbf{T}_j &\equiv& \mathbf{K}_j+\frac{\mathbf{J}-\mathbf{J}'}{2} \\ \mathbf{T}'_{j-1} &\equiv& \mathbf{K}_{j-1}-\frac{\mathbf{J}-\mathbf{J}'}{2} \\ \mathbf{T}'_j &\equiv& \mathbf{K}_j-\frac{\mathbf{J}-\mathbf{J}'}{2} \end{array}\right. \quad (25)$$

where $\mathbf{J}, \mathbf{K}_{j-1}, \mathbf{J}'$ and $\mathbf{K}_j$ are the right sides of equations (17), (18), (19) and (20) respectively. Noting that (21) means (17) + (19) = (18) + (20), we can easily confirm that (25) satisfies (17) through (20).

Finally, we consider twist constraints (3) around $\mathbf{V}_j$. Let ν^k $(k = 0, 1, 2)$ be displacement factors at three vertices of $\mathbf{W}$ type around $\mathbf{V}_j$. Noting that $\mu = -\frac{1}{2}$, we have three twist constraints

$$\frac{\mathbf{S}_j^{k+1}+\mathbf{S}_j^{k-1}}{2} = \frac{\nu^k}{3}\mathbf{V}_j+\left(\frac{4}{3}-\frac{\nu^k}{3}\right)\mathbf{P}_j^k-\frac{1}{3}\mathbf{Q}_j^k \quad (k=0,1,2) \quad (26)$$

which can be uniquely solved for $\mathbf{S}^k$ $(k = 0, 1, 2)$ as

$$\mathbf{S}_j^k = \left\{\frac{\nu^{k-1}}{3}\mathbf{V}_j+\left(\frac{4}{3}-\frac{\nu^{k-1}}{3}\right)\mathbf{P}_j^{k-1}-\frac{1}{3}\mathbf{Q}_j^{k-1}\right\}$$
$$-\left\{\frac{\nu^k}{3}\mathbf{V}_j+\left(\frac{4}{3}-\frac{\nu^k}{3}\right)\mathbf{P}_j^k-\frac{1}{3}\mathbf{Q}_j^k\right\}$$
$$+\left\{\frac{\nu^{k+1}}{3}\mathbf{V}_j+\left(\frac{4}{3}-\frac{\nu^{k+1}}{3}\right)\mathbf{P}_j^{k+1}-\frac{1}{3}\mathbf{Q}_j^{k+1}\right\} \quad (27)$$

4.3 Summary of computation of Bézier control points

We have described our procedure to compute control points so far, which can be summarized as follows:

1. For each cell, compute intermediate points $\mathbf{D}_j$ and $\mathbf{E}_j$ by (8) and (12) respectively.
2. At each vertex $\mathbf{U}$, compute the displacement factor λ and surrounding control points $\mathbf{A}_j$, $\mathbf{B}_j$ and $\mathbf{R}_j$ by (10) and (11).
3. At each 3-valent vertex $\mathbf{V}_j$, compute the displacement factor μ, control points $\mathbf{P}_j^k$ and the vertex itself by (14).
4. At each 4-valent vertex $\mathbf{W}$, compute the displacement factor ν and surrounding control points $\mathbf{Q}_j$, $\mathbf{T}_j$, etc. by (24), (23) and (25). The vertex itself is determined by (15).
5. At each 3-valent vertex $\mathbf{V}_j$, compute $\mathbf{S}_j^k$ determining twists by (27).

5 Experiments

The proposed algorithm was implemented and validated through experiments. The input meshes were generated by a volumetric triangulation method[12] in different resolutions from unorganized 3D point sets captured by NRCC range sensor.

Figure 5 shows the original input mesh with 290 triangles and several views of the generated surface. All the surfaces are rendered with flat shading. Though the blend ratio a may vary vertex by vertex, we here set them to an uniform value: $a = 0.7$. The shift parameter α was fixed to 0.8. It was proved that the bending parameter c in (24) scarcely affects the surface shape for the values $c \in [0.5, 2.0]$. We therefore uniformly set $c = 1.0$. The CPU time required for computation of the control points was within a second on a Sun Ultra 2 workstation.

The shape of the resulting surface can be controlled by varying the blend ratio a. Figure 6 shows the generated surfaces and corresponding control nets for two different values of a: 0.15 and 0.9. It can be observed that the density of the control points is high in the neighborhood of the edges of the original triangular mesh for a small a value, which produces a flat shaped surface. On the other hand, the control points distributes more uniformly for large a and the round shape results.

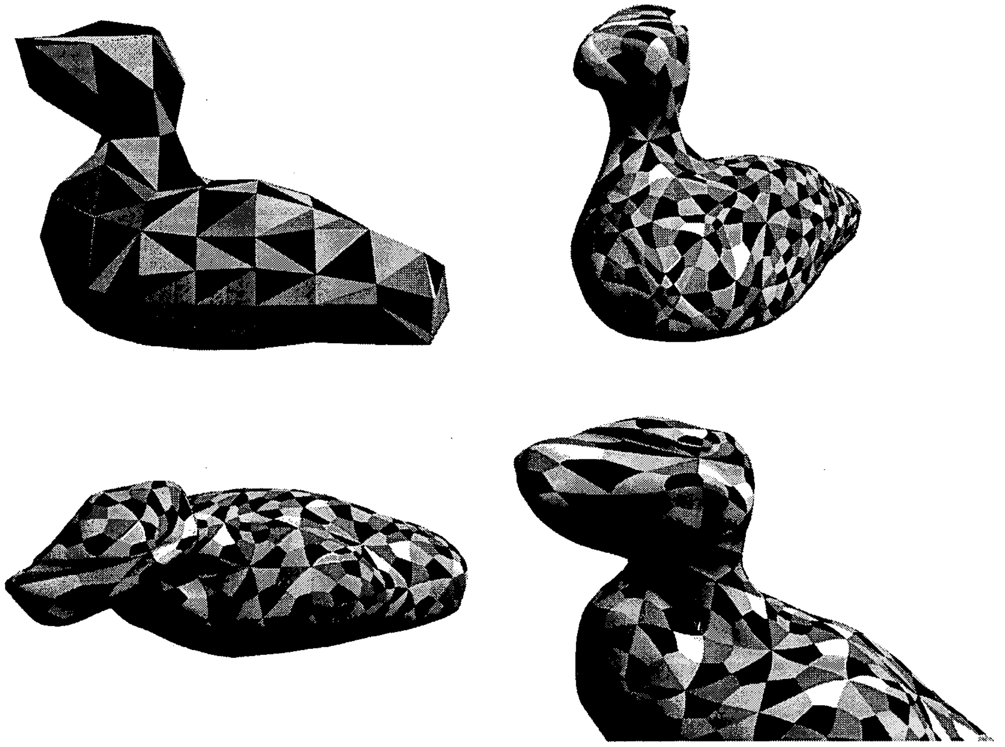


Figure 5. The original mesh (duck) with 290 faces and generated surface: $a = 0.7,\ \alpha = 0.8,\ c = 1.0$

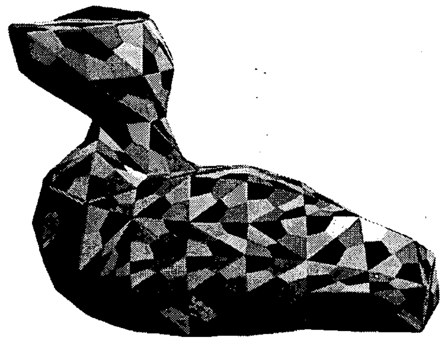
Three more examples are displayed in figures 7, 8 and 9.

6 Conclusions

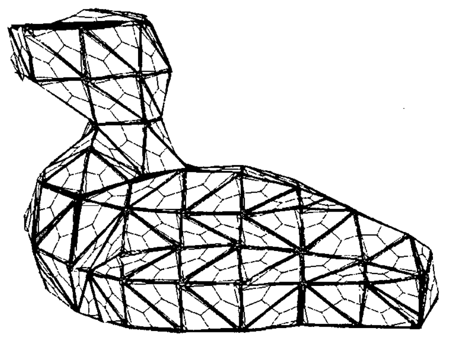
A new algorithm for constructing a tangent plane continuous spline surface has been presented. This algorithm can create G^1 surfaces with bicubic Bézier rectangles over closed triangular meshes of arbitrary topological type. Each triangle is partitioned into three quadrilaterals and rectangular Bézier patches are assigned to them. Because the fact that all the original mesh faces are triangular is fully exploited, the degree of each patch is optimized to three while the more general approach[9] requires degree four patches. The construction procedure is affine invariant, that is, an affinely transformed G^1 surface can be obtained by applying the same transformation to the control points. In addition, since all the control points are computed locally, the algorithm is very efficient. The surface shape can be intuitively controlled by changing values of blend ratio.

An important problem not solved yet is to extend this method so that the meshes with boundary or non-closed meshes are allowed. The surface should interpolate the boundary curve while satisfying some boundary conditions specified in terms of tangent functions.

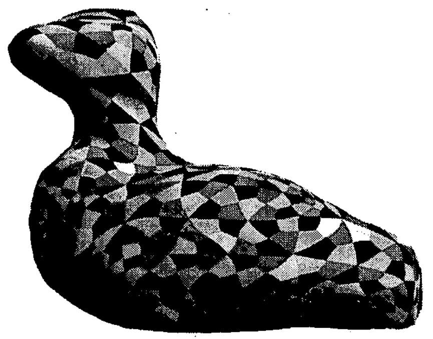
In the future, we plan to apply our method to automatic model building from unorganized range data[2, 5, 13]. In this case, mesh vertices, generating points, blend ratios, shift parameters and intermediate points are adjusted through some optimization process so that the resulting surface well approximates the given data. In addition, some procedure which adaptively splits or merges mesh faces according to the distribution of the input data points should be incorporated in order to attain the best approximation with small storage. The proposed algorithm would provide an initial guess for this process.



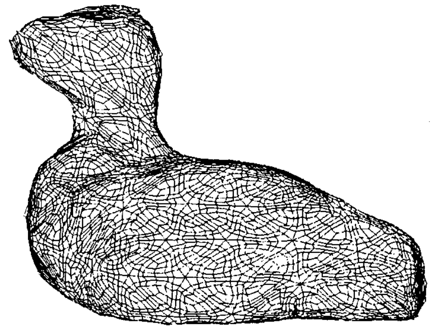
(a) Generated surface: $a = 0.15$, $\alpha = 0.8$, $c = 1.0$



(b) Control net for (a)



(c) Generated surface: $a = 0.9$, $\alpha = 0.8$, $c = 1.0$



(d) Control net for (c)

Figure 6. Generated surfaces and control nets for diferrent values of blend ratio.

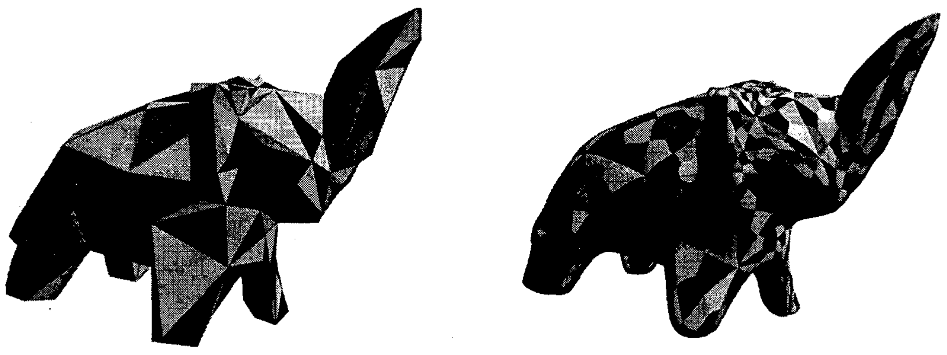


Figure 7. The original mesh (elephant) with 352 faces and generated surface: $a = 0.6$, $\alpha = 0.8$, $c = 1.0$

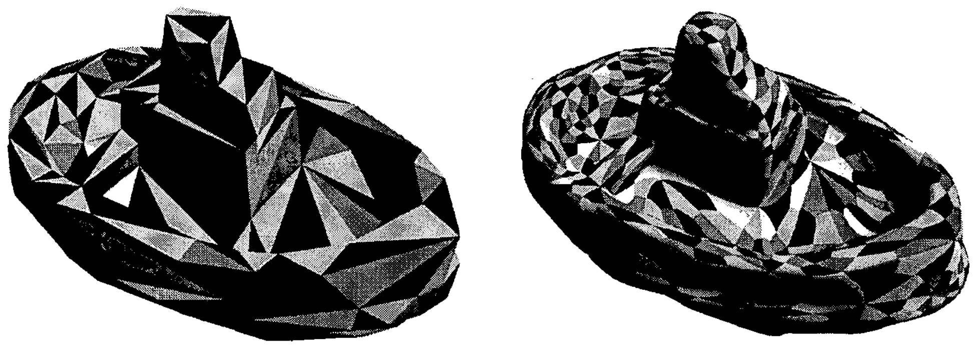


Figure 8. The original mesh (boat) with 372 faces and generated surface: $a = 0.5,\ \alpha = 0.8,\ c = 1.0$

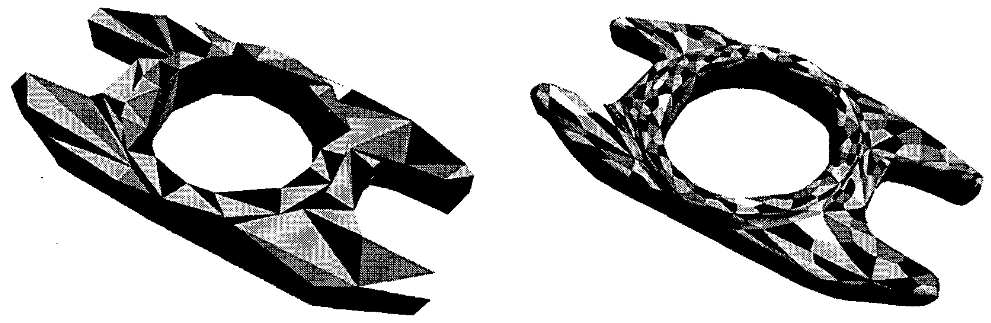


Figure 9. The original mesh (grapple) with 256 faces and generated surface: $a = 0.5,\ \alpha = 0.8,\ c = 1.0$

Acknowledgments

The first author would like to thank Jacques Domey, the group leader of Visual Information Technology Group, for giving the author an invaluable opportunity to stay at VIT and for making this work possible. We thank Pierre Gariépy for his support in computational facilities.

A Proof of equation (9)

Substituting (8) into (9) and rearranging, we have

$$\begin{aligned}
&\frac{\mathbf{D}_{j-1}+\mathbf{D}_{j+1}}{2} \\
&= \mathbf{U} + \frac{\alpha}{2n}\sum_{k=0}^{n-1}\left\{\cos\frac{2\pi(j-1-k)}{n} + \cos\frac{2\pi(j+1-k)}{n}\right\}\mathbf{C}_k \\
&= \mathbf{U} + \frac{\alpha}{n}\sum_{k=0}^{n-1}\cos\frac{2\pi}{n}\cos\frac{2\pi(j-k)}{n}\mathbf{C}_k \\
&= \left(1-\cos\frac{2\pi}{n}\right)\mathbf{U} \\
&\quad + \cos\frac{2\pi}{n}\left(\mathbf{U} + \frac{\alpha}{n}\sum_{k=0}^{n-1}\cos\frac{2\pi(j-k)}{n}\mathbf{C}_k\right) \\
&= \left(1-\cos\frac{2\pi}{n}\right)\mathbf{U} + \cos\frac{2\pi}{n}\mathbf{D}_j
\end{aligned}$$

where the following trigonometric identity is used:

$$\cos\theta + \cos\phi = 2\cos\frac{\theta+\phi}{2}\cos\frac{\theta-\phi}{2}.$$

■

B Verification of the solution (23)

Substituting (23) into the left side of (21) and using (24), we have

$$
\begin{aligned}
&\left(1-\frac{\lambda+\lambda'}{6}+\frac{2}{3}\nu\right)\mathbf{Q}_{j-1}+\left(1-\frac{\lambda+\lambda'}{6}-\frac{2}{3}\nu\right)\mathbf{Q}_j \\
&= \left(1-\frac{\lambda+\lambda'}{6}\right)(\mathbf{Q}_{j-1}+\mathbf{Q}_j)-\frac{2}{3}\nu(\mathbf{Q}_j-\mathbf{Q}_{j-1}) \\
&= \frac{1}{10-\lambda-\lambda'}\left[\left(1-\frac{\lambda+\lambda'}{6}\right)\{2(5-\lambda)\mathbf{B}_j\right. \\
&\qquad +2(5-\lambda')\mathbf{B}'_j-4c(\lambda-\lambda')(\mathbf{P}_j-\mathbf{P}_{j-1})\} \\
&\qquad \left.-\frac{2}{3}c(\lambda-\lambda')\left\{\frac{1}{c}\mathbf{B}_j-\frac{1}{c}\mathbf{B}'_j+4(\mathbf{P}_j-\mathbf{P}_{j-1})\right\}\right] \\
&= \frac{1}{10-\lambda-\lambda'}\left[\left\{\left(\frac{10-\lambda-\lambda'}{6}-\frac{2}{3}\right)2(5-\lambda)\right.\right. \\
&\qquad \left.-\frac{2}{3}(-(5-\lambda)+(5-\lambda'))\right\}\mathbf{B}_j \\
&\qquad +\left\{\left(\frac{10-\lambda-\lambda'}{6}-\frac{2}{3}\right)2(5-\lambda')\right. \\
&\qquad \left.+\frac{2}{3}(-(5-\lambda)+(5-\lambda'))\right\}\mathbf{B}'_j \\
&\qquad \left.-\left\{\left(1-\frac{\lambda+\lambda'}{6}\right)+\frac{2}{3}\right\}4c(\lambda-\lambda')(\mathbf{P}_j-\mathbf{P}_{j-1})\right] \\
&= \frac{1}{10-\lambda-\lambda'}\left[\left\{(10-\lambda-\lambda')\frac{5-\lambda}{3}\right.\right. \\
&\qquad \left.-\frac{2}{3}((5-\lambda)+(5-\lambda'))\right\}\mathbf{B}_j \\
&\qquad +\left\{(10-\lambda-\lambda')\frac{5-\lambda'}{3}-\frac{2}{3}((5-\lambda)+(5-\lambda'))\right\}\mathbf{B}'_j \\
&\qquad \left.-\left(\frac{5}{3}-\frac{\lambda+\lambda'}{6}\right)4c(\lambda-\lambda')(\mathbf{P}_j-\mathbf{P}_{j-1})\right] \\
&= \left(\frac{5-\lambda}{3}-\frac{2}{3}\right)\mathbf{B}_j+\left(\frac{5-\lambda'}{3}-\frac{2}{3}\right)\mathbf{B}'_j \\
&\qquad -\frac{2}{3}c(\lambda-\lambda')(\mathbf{P}_j-\mathbf{P}_{j-1}) \\
&= \left(1-\frac{\lambda}{3}\right)\mathbf{B}_j+\left(1-\frac{\lambda'}{3}\right)\mathbf{B}'_j-\frac{2}{3}\nu(\mathbf{P}_j-\mathbf{P}_{j-1}).
\end{aligned}
$$

Similarly, substituting (23) into the left side of (22) and using (24), we have

$$
\begin{aligned}
&(1-\nu)\mathbf{Q}_{j-1}+(1+\nu)\mathbf{Q}_j \\
&= \mathbf{Q}_{j-1}+\mathbf{Q}_j+\nu(\mathbf{Q}_j-\mathbf{Q}_{j-1}) \\
&= \frac{1}{10-\lambda-\lambda'}\left[2(5-\lambda)\mathbf{B}_j+2(5-\lambda')\mathbf{B}'_j\right. \\
&\qquad -4c(\lambda-\lambda')(\mathbf{P}_j-\mathbf{P}_{j-1}) \\
&\qquad \left.+c(\lambda-\lambda')\left\{\frac{1}{c}\mathbf{B}_j-\frac{1}{c}\mathbf{B}'_j+4(\mathbf{P}_j-\mathbf{P}_{j-1})\right\}\right] \\
&= \frac{1}{10-\lambda-\lambda'}[\{2(5-\lambda)+(\lambda-\lambda')\}\mathbf{B}_j \\
&\qquad +\{2(5-\lambda')-(\lambda-\lambda')\}\mathbf{B}'_j] \\
&= \mathbf{B}_j+\mathbf{B}'_j.
\end{aligned}
$$

■

References

[1] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH 96*, pages 221–227, 1996.

[2] M. Eck and H. Hoppe. Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type. In *Proc. SIGGRAPH 96*, pages 325–334, 1996.

[3] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design (2nd Ed.).* Academic Press, 1990.

[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface Reconstruction from Unorganized Points. *ACM Computer Graphics*, 26(2):71–78, July 1992.

[5] V. Krishnamurthy and M. Levoy. Fitting Smooth Surfaces to Dense Polygon Meshes. In *Proc. SIGGRAPH 96*, pages 313-324, 1996.

[6] C. Loop. A G^1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, 11(3):303–330, 1994.

[7] C. Loop. Smooth Spline Surfaces over Irregular Meshes. In *Proc. SIGGRAPH 94*, pages 303–310, 1994.

[8] C. Loop and T. DeRose. Generalized B-spline Surfaces of Arbitrary Topology. *ACM Computer Graphics*, 24(4):343–356, Aug 1990.

[9] J. Peters. Biquartic C^1-surface splines over irregular meshes. *Computer Aided Design*, 27(12):895–903, 1995.

[10] L. Piegl and W. Tiller. *The NURBS Book*. Springer, 1995.

[11] U. Reif. Biquadratic G-spline surfaces. *Computer Aided Geometric Design*, 12:193–205, 1995.

[12] G. Roth and E. Wibowoo. An Efficient Volumetric Method for Building Closed Triangular Meshes from 3-D Image and Point Data. In *Proc. Graphics Interface 97*, pages 173–180, 1997.

[13] A. J. Stoddart and M. Baker. Reconstruction of Smooth Surfaces with Arbitrary Topology Adaptive Splines. In *Proc. 5th European Conference on Computer Vision*, volume 2, pages 241–254, 1998.