ELSEVIER

Computer Aided Geometric Design 19 (2002) 89–112

COMPUTER AIDED GEOMETRIC DESIGN

www.elsevier.com/locate/comaid

Hierarchical multiresolution reconstruction of shell surfaces

Chandrajit L. Bajaj [a,1], Guoliang Xu [b,2], Robert J. Holt [c,*], Arun N. Netravali [c]

[a] *Department of Computer Sciences, University of Texas, Austin, TX 78712, USA*

[b] *State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences, Beijing, PR China*

[c] *Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974, USA*

Received April 2000; revised July 2001

Abstract

We present an adaptive, hierarchical Hh-multiresolution reconstruction algorithm to model shell surface objects from a matched pair of triangulated surfaces. Shell surfaces are an interval of contours of trivariate functions with prismatic support. In the H-direction, a hierarchical representation of the scaffold is constructed. For any adaptively extracted scaffold from the hierarchy, a sequence of functions in the h-direction (regularly subdivided mesh) is constructed so that their contours approximate the input shell to within a given error ε. The shell surfaces can be made to capture sharp curve creases on the shell while being C^1 smooth everywhere else. Using an interval of iso-contours of smooth trivariate spline functions, rather than a pair of inner and outer surface splines, one avoids the need for interference checks between the inner and outer surface boundaries. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Geometric modeling; Level of detail algorithms; Mesh generation

1. Introduction

Many human manufactured and naturally occurring objects have shell-like structures, that is, the object bodies consist of surfaces with thickness. We call such surfaces *shell surfaces*. The problem of constructing smooth approximations to shell surface objects

* Corresponding author.
E-mail address: rjh@research.bell-labs.com (R.J. Holt).
[1] Research supported in part by NSF grants CCR 9732306, KDI-DMS-9873326, and ACI-9982297.
[2] Research supported in part by NSFC grant 19671081.

arises in creating geometric models such as airfoils, tin cans, shell canisters, engineering castings, sea shells, the earth's outer crust, and the human skin, to name just a few.

1.1. Problem description

In engineering, shell structures are often analyzed by finite element methods (see (Bernadou and Boisserie, 1982; Bucalem and Bathe, 1997; Cirak et al., 1999; Mollmann, 1981; Szabo and Babuska, 1991)). In these analyses, the shell is often assumed to be uniform in thickness for simplicity, hence the output is often a triangulation that represents the mid-surface of the shell. More accurate finite element analysis of shells uses volume elements, such as hexahedral (see (Liu et al., 1998)) or pentahedral (see (Szabo and Babuska, 1991)). In these cases, the output may be a matched triangulation pair. Bajaj and Xu (1999) also present schemes for obtaining matched triangulation pairs for varied scattered and dense surface data inputs. In this paper, our aim is to reconstruct smooth shells from matched triangulation pairs. However, we do not assume that the shell is uniform in thickness; instead, we assume we are given two triangulations that represent the boundaries of the shell. These triangulations can be obtained by offsetting the mid-surface triangulation in the normal direction with varying thickness. In the model (such as airfoil, arched roof and dam etc.) construction, we must often respect the geometric data and therefore cannot assume the shell is uniform in thickness. Hence, our problem may be described as follows.

Problem description. As input we are given a matched triangulation pair $\mathcal{T} = \{\mathcal{T}^{(0)}, \mathcal{T}^{(1)}\}$ with attached normals at each vertex, which presents a linearization of the inner and outer boundary surfaces of a shell domain; also, we are given an error control tolerance $\varepsilon > 0$. The goal is to reconstruct hierarchical multiresolution smooth shell surfaces whose bounding surfaces provide approximations of $\mathcal{T}^{(0)}$ and $\mathcal{T}^{(1)}$, respectively, with errors no larger than ε.

The hierarchical scheme is comprised of multiresolutions in two directions. The terminology *Hh-multiresolution* we use is borrowed from hp-finite element analysis (see (Szabo and Babuska, 1991)), where their "h" elements denote the mesh size and their "p" elements denote the degree of the shape functions on each mesh element. Here the H-direction multiresolution is a level-of-detail (LOD) representation of the pair of (irregular) triangular meshes, and the h-direction multiresolution is the regular subdivision of each of the triangle pairs. In the geometric modeling problem, using high degree polynomials in general leads to surfaces containing pronounced waves and also increases the computational costs. Hence, we use triangular cubic spline functions on the regularly partitioned triangles.

1.2. Prior solution approaches

Traditionally, a thin shell has been often treated as a single surface (see (Bernadou and Boisserie, 1982; Ketchum and Ketchum, 1997)) by taking the mid-surface of the shell or assuming that the thickness is zero. These treatments work fine in the cases where the thickness has little effect on the solution. However, if the thickness is not small enough or it varies significantly, using a single surface to represent the shell will not be accurate (see, e.g., Figs. 5.3 and 6.2). Therefore, two boundaries of the shell as well as surfaces in between need to be constructed. Of course, one could solve the proposed geometric modeling problem by using classical or existing methods (see, e.g., (Farin, 1990; Hoschek and Lasser, 1993; Piegl and Tiller, 1997)) of parametric surface splines to construct individual boundary surfaces as well as mid-surfaces of the shell boundaries. However, the independent construction of each surface not only increases tremendously the space and time costs, but also fails to guarantee that these surfaces are always separate. In particular, post-local and/or -global interactive surface modification requires extremely cumbersome surface–surface interference checks to be performed in order to preserve geometric model consistency.

To the authors' knowledge, the reconstruction of shell structures by a unified approach is a new area. In an earlier paper (Bajaj and Xu, 1999), we proposed an adaptive approach in which the shell surface is defined by the contours of a single trivariate function F. This function is defined on a collection of triangular prisms (prism complex, or scaffold) in $\mathbb{R}^3$, such that it is C^1 and its contour $F(x,y,z)=\alpha$ for any $\alpha \in (-1,1)$ provides a smooth mid-surface with $F(x,y,z)=-1$ and $F(x,y,z)=1$ as the inner and outer boundaries of the shell structure. This implicit method is shown to be efficient and superior to the parametric method mentioned above in which the boundaries and mid-surfaces are all incorporated into one trivariate function.

1.3. Our approach

In this paper, we extend the reconstruction method of Bajaj and Xu (1999) to achieve (a) hierarchical Hh-multiresolution, (b) ε-bounded approximations, and (c) the ability to capture sharp curve creases while being C^1 smooth everywhere else. To achieve adaptive multiresolution representations in the H-direction, a hierarchical presentation of the prism scaffold is constructed. For each extracted scaffold from the hierarchy, a sequence of functions (h-direction) is constructed, using triangular splines on regularly subdivided triangular prisms, such that the input data is approximated to within the allowable error ε. To get an adaptive reconstruction, combinations of different levels in both the H- and h-directions are allowed.

The remainder of the paper is organized as follows. In Section 2 we introduce some notation used for describing our algorithm. We then outline in Section 3 the complete algorithm steps for solving the proposed problem. These steps are detailed in the sections that follow. Two heavy tasks in this algorithm, that are tackled in Sections 4 and 5, are the geometric construction of the hierarchical scaffold and the C^1 function construction over the scaffold. The hierarchical construction of the prism scaffold is the same in nature as that of the hierarchical construction of a unique triangulation (hence some steps are detailed in the Appendices A and B), but with some adjustments to fit our shell triangulation problem. The C^1 construction in Section 5 is basically a local interpolation approach, that utilizes mainly the tools of one-dimensional Hermite interpolation, one-dimensional B-splines, two-dimensional triangular B-splines, and transfinite interpolation. The problems handled by Sections 6 and 7 are relatively lighter. In Section 6 we consider the problem of preserving sharp features while constructing the smooth shell surfaces, and in Section 7 we evaluate and display the shell surfaces. Section 8 concludes the paper with additional examples.

2. Notation

We assume $\mathcal{T}^{(0)}$ and $\mathcal{T}^{(1)}$ are orientable. For each vertex pair $V_i = \{V_i^{(0)}, V_i^{(1)}\}$ with attached normal pair $\{N_i^{(0)}, N_i^{(1)}\}$, we assume

$$\left[V_i^{(1)} - V_i^{(0)}\right]^{\mathrm{T}} N_i^{(s)} > 0, \quad s = 0, 1.$$

This ensures that points in the outer layer are roughly in the same direction from the corresponding points in the inner layer as the normals. With this convention the normals to both the inner and outer surfaces are outward-pointing normals, in contrast to the more common convention where the normals to the inner surface are inward-pointing. For each triangle pair $[V_i V_j V_k]$, we further assume

$$\left[V_i^{(0)} V_j^{(0)} V_k^{(0)}\right] \cap \left[V_i^{(1)} V_j^{(1)} V_k^{(1)}\right] = \emptyset.$$

Our trivariate function F for constructing the shell is piecewise defined on a collection of prisms. Let $[V_i V_j V_k]$ be a triangle pair. Then the prism, denoted by P_{ijk}, for $[V_i V_j V_k]$ is a volume in $\mathbb{R}^3$ enclosed by the surfaces H_{ij}, H_{jk}, and H_{ki} (see Fig. 2.1), where H_{lm} is a ruled surface defined by V_l and V_m as follows:

$$H_{lm} = \left\{p\colon\ p = b_1 v_l(\lambda) + b_2 v_m(\lambda),\ b_1 + b_2 = 1,\ \lambda \in \mathbb{R}\right\}$$

with $v_i(\lambda) = V_i^{(0)} + \lambda N_i$, $N_i = V_i^{(1)} - V_i^{(0)}$. We will wish to describe points within these prisms in terms of the triangle vertex pairs, as a type of “shell barycentric coordinate”. To this end we can explicitly represent the prism P_{ijk} as the volume given by

$$P_{ijk}(I) = \left\{p\colon\ p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda),\ b_1 + b_2 + b_3 = 1,\ b_l \geqslant 0,\ \lambda \in I\right\},$$

where I is a specified interval. This interval contains $[0, 1]$ and is usually larger, so that each prism P_{ijk} contains the triangle pair $[V_i^{(0)} V_j^{(0)} V_k^{(0)}]$ and usually extends past its faces, as illustrated in Fig. 2.1. We call (b_1, b_2, b_3, λ) the P_{ijk}-coordinate of

$$p = p_{ijk}(b_1, b_2, b_3, \lambda) = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda).$$

For each $\lambda \in I$,

$$T_{ijk}(\lambda) := \left\{p\colon\ p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda),\ b_1 + b_2 + b_3 = 1,\ b_l \geqslant 0\right\}$$

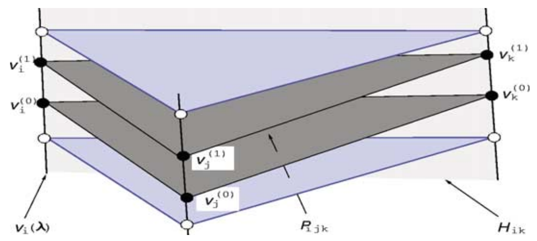


Fig. 2.1. The volume prism cell P_{ijk}, the face $H_{ik}(t, \lambda)$ and the edge $v_i(\lambda)$ defined by a triangle pair $[V_i V_j V_k]$.

defines a triangle. To ensure that this triangle is non-degenerate, λ is confined to lie in a certain interval I_{ijk}. This interval is computed as follows.

Let

$$p_{ijk}^{(l)}(\lambda) = \det\bigl[n_l, v_j(\lambda) - v_i(\lambda), v_k(\lambda) - v_i(\lambda)\bigr], \quad l = i, j, k.$$

Assume

$$p_{ijk}^{(l)}(\lambda) > 0, \quad \forall \lambda \in [0,1],\ l = i, j, k. \tag{2.1}$$

Consider the real numbers $\lambda_1, \ldots, \lambda_s$ ($s \leqslant 6$) that solve one of these three equations of degree 2: $p_{ijk}^{(l)}(\lambda) = 0$, $l = i, j, k$, and define $a = \max(-\infty, \{\lambda_l\colon \lambda_l < 0\})$, $b = \min(+\infty, \{\lambda_l\colon \lambda_l > 1\})$, and $I_{ijk} = (a, b)$. Then I_{ijk} is the largest interval containing $[0,1]$ such that $P_{ijk}(I_{ijk})$ is non-degenerate. To show this fact, note that a triangle $T_{ijk}(\lambda)$ is non-degenerate if and only if

$$n_l^{\mathrm{T}}\bigl[v_j(\lambda) - v_i(\lambda)\bigr] \times \bigl[v_k(\lambda) - v_i(\lambda)\bigr] = p_{ijk}^{(l)}(\lambda) > 0, \tag{2.2}$$

$l = i, j, k$, where $\times$ denotes the cross product of two vectors. The assumption (2.1) implies that $[0,1] \subset I$. Since $p_{ijk}^{(l)}(0) > 0$ and $p_{ijk}^{(l)}(1) > 0$, for $l = i, j, k$, then $p_{ijk}^{(l)}(\lambda) > 0$ for $\lambda \in (a, b)$ and $l = i, j, k$. Since $p_{ijk}^{(l)}(a) = 0$ for $l = i$ or $l = j$ or $l = k$ if $a > -\infty$, a is the infimum of the interval of λ that contains $[0,1]$ and makes (2.2) hold. Similarly, b is the supremum of such an interval. Therefore I_{ijk} is the largest interval such that $P_{ijk}(I_{ijk})$ is non-degenerate.

We call the union of all $P_{ijk}(I_{ijk})$ a prism scaffold. For the input triangulation pair, the corresponding scaffold, denoted as $\mathcal{S}_0$, will be the finest level in our hierarchical representation of the scaffold. Note that the triangulation (we always mean the matched triangulation pair) and the scaffold correspond closely. The vertex V_i, edge $[V_i V_j]$ and triangle $[V_i V_j V_k]$ of the triangulation correspond to the edge $v_i(\lambda)$, face H_{ij} and prism P_{ijk} of the scaffold, respectively. Hence, any operation conducted on the triangulation implies the same on the scaffold. For instance, removing a vertex pair from the triangulation and then re-triangulating implies removing an edge from the scaffold and then “re-meshing” the prism scaffold. These operations are performed in building the hierarchical representation of the scaffold in Section 4.

Given a P_{ijk}-coordinate for a point, it is straightforward to compute its coordinates in the xyz system. However, the inverse is not trivial, since the transforms between them are nonlinear. For a given $p \in \mathbb{R}^3$, we determine $(b_1, b_2, b_3, \lambda)^{\mathrm{T}}$ such that

$$p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda), \quad b_1 + b_2 + b_3 = 1. \tag{2.3}$$

It follows from (2.3) that we have

$$p - v_k(\lambda) = \bigl[v_i(\lambda) - v_k(\lambda), v_j(\lambda) - v_k(\lambda)\bigr][b_1,\ b_2]^{\mathrm{T}}.$$

Therefore,

$$\det\bigl[p - v_k(\lambda), v_i(\lambda) - v_k(\lambda), v_j(\lambda) - v_k(\lambda)\bigr] = 0. \tag{2.4}$$

The left-hand side of (2.4) is a polynomial of degree 3 in λ. Upon solving this equation for λ, we choose the root such that the solution (b_1, b_2, b_3) of (2.3) satisfies $b_i \geqslant 0$, $\sum b_i = 1$.

Whenever it is necessary to address the functions that are defined on the level t scaffold (H-direction), the notation $F^{(t)}$ is used. The notation F_σ will be used to address the level σ function in the h-direction.

3. Algorithm outline

The hierarchical construction algorithm of the shell structures is comprised of two main phases: the hierarchical construction of the scaffold and of the function over the scaffold. This section gives the algorithm pipeline, with the details of the algorithm provided in the sections that follow.

Step 1. Construct a C^1 function on the scaffold $\mathcal{S}_0$.

The finest level scaffold $\mathcal{S}_0$ is built on the input matched triangulation pair. On this scaffold, a C^1 function $F^{(0)}$ is constructed (see Section 5.1). This function is regarded as exact when constructing other functions at other resolutions.

Step 2. Hierarchical representation of scaffold.

This step constructs a directed acyclic graph (DAG) for the levels of detail of the scaffold. This DAG is built based on the algorithm in (de Berg and Dobrindt, 1998; De Floriani et al., 1999), with changes to the vertex removal criterion and hole re-triangulation method (see Section 4). Having such a DAG, we are able to travel from a fine level to a coarse one or vice versa, and extract a required scaffold satisfying a given control error by combining different levels.

Step 3. Adaptive scaffold extraction.

For the given control parameters, extract a required scaffold from the DAG that satisfies the given condition (see Section 4.1).

Step 4. Face data construction.

For each face of the prisms in any level, a C^2 function and C^1 gradient on the face are constructed. All these data form a list. In the DAG structure, each prism should have three pointers that point to the corresponding face data (see Section 5.3). Having this data, we are able to construct C^1 functions on any extracted scaffold.

Step 5. Construct trivariate splines in each prism.

For the given control error ε and the selected scaffold, construct a sequence of C^1 trivariate splines F_σ, $\sigma = 1, 2, \ldots, \Sigma$, so that

$$S_\alpha^{(\sigma)} = \{p\colon\ F_\sigma(p) = \alpha,\ \alpha \in [-1, 1]\}, \quad \sigma = 1, 2, \ldots, \Sigma,$$

are smooth surfaces and $S_{-1}^{(\Sigma)}$ and $S_1^{(\Sigma)}$ are ε error-bounded approximations of the inner and outer boundary surfaces of the input shell, respectively. In the process of this construction certain *curve creases* are tagged and captured. This step is described in detail in Sections 5.4 and 6.

Step 6. Evaluate and display the shell surface.

See Section 7 for details.

4. Hierarchical representation of prism scaffold

The hierarchical representation of the scaffold is a sequence $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_k$ of scaffolds, from the finest level to the coarsest. To construct the hierarchical representation of the scaffold, we perform recursively a vertex removal procedure to form a sequence $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_k$ of matched triangulation pairs, where $\mathcal{T}_0 = \mathcal{T}$. The policy of the vertex removal is adopted from (de Berg and Dobrindt, 1998; De Floriani et al., 1999). That is, if one vertex is selected to be removed at level t, then its neighbor vertices at the same level may not be removed. Hence any two vertices in the set of vertices that are going to be removed are disconnected (see Fig. 4.1). The next level of triangulation is obtained by re-triangulating holes that are left when the vertices are removed.

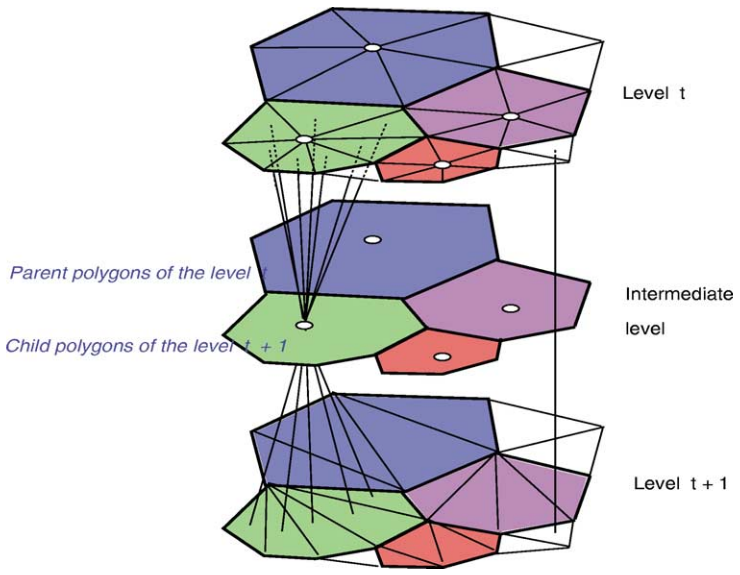

Level t
Parent polygons of the level t
Intermediate level
Child polygons of the level t + 1
Level t + 1

Fig. 4.1. The vertices with circles at the top level are the ones that are removed at level t. The star-shaped polygons, that are shown in the middle of the figure and obtained by removing the selected vertices, are re-triangulated as shown at the bottom. Newly formed prisms at level $t + 1$ are linked to those at level t by arcs through the intermediate level. The unchanged prisms are linked directly.

The hierarchy is stored as a directed acyclic graph (DAG), whose nodes correspond to the prisms of $\mathcal{S}_0$ up to $\mathcal{S}_k$. The leaf nodes correspond to the prisms of $\mathcal{S}_0$. Between the levels t and $t+1$, there is an intermediate level that corresponds to the removed vertices of level t. There is an arc from the star-shaped polygon that is formed when a vertex is removed, to every triangle in level t around the vertex, and to every triangle in level $t+1$ formed by the re-triangulation of the star-shaped polygon. A polygon at the intermediate level between levels t and $t+1$ is called the *parent polygon* of those prisms at level t that linked to it, and it is also called the *child polygon* of those prisms at level $t+1$ that are linked to it. Unchanged triangles between two levels are linked directly by arcs. These descriptions are illustrated by Fig. 4.1.

Some data must be stored along with the DAG. First is the vertex pair list *VertexList*, which is fixed and does not change during the construction of the DAG. Another list is *FaceList*, that of the faces of the prism, which is incremental. The initial list is that of the faces of $\mathcal{S}_0$. Each entry of *FaceList* contains the information of the C^2 function and the C^1 gradient on that face (see Section 5.3). When new prisms are produced, the new faces are added to this list. In the DAG, three pointers that point to the three faces of each prism must be stored. Having this information allows us to construct later C^1 functions within each prism cell.

To achieve our goal of building the hierarchical representation of the scaffold, there are two points that need to be addressed. One is the vertex removal criterion, and the other is the re-triangulation. These steps are detailed in Appendices A and B.

4.1. *Adaptive extraction of shell surface support*

It is obvious that simply taking a certain level of the scaffold from the hierarchy does not have the adaptive nature. Therefore, it is necessary to combine different level scaffolds to form an adaptive one. The extraction algorithm in (de Berg and Dobrindt, 1998; De Floriani et al., 1999) can be altered to serve our purpose. From the construction of the DAG we know that each prism in any level has a grade that measures the normal variation. We shall use this grade to control the scaffold extraction for a given control value $g \in [0, \pi/2)$ of the normal variation. To describe the extraction algorithm precisely, we introduce some more notation. Let P be a prism of level t. Then we denote by $G_t(P)$ the collection of all prisms at level t that are in the same child polygon as P, and by $G_t^c(P)$ the collection of all prisms at level $t-1$ that are linked to child polygons of P. Let $Sub_t(P)$ be the collection of all prisms in the levels $t, t-1, \ldots, 0$, that are linked directly or indirectly through intermediate nodes to the prisms in $G_t^c(P)$. That is, $Sub_t(P)$ consists of the prisms in the sub-DAG starting with $G_t(P)$. Then the algorithm for extracting the scaffold can be described by the following C language style pseudo-code:

```
$Q_k = \mathcal{S}_k$;                 /* put all the prisms in $\mathcal{S}_k$ to $Q_k$*/
for ($t = k$;  $t > 0$;  $t --$) {
    $Q_{t-1}$ = NULL;
    while ($Q_t \neq$ NULL ) {
      $P = Q_t[0]$;
      if ($G_t(P) \not\subset Q_t$) {
          accept all the prisms in $G_t(P)$;
```

```
            }   else {
                if (Grade($p$) $\leqslant g$ for all $p \in Sub_t(P)$ ) {
                    accept all the prisms in $G_t(P)$
                } else {
                    append to the end of $Q_{t-1}$ all the prisms in $G_t^c(P)$
                }
            }
            remove from $Q_t$ all the prisms that in $G_t(P)$;
        }
    }
    if ($Q_0 \neq$ NULL) {
        accept all the prisms in $Q_0$;
    }.
```

5. Construction of C^1 trivariate functions on hierarchy

The C^1 functions on the hierarchy are constructed in three steps: (a) A C^1 function $F^{(0)}$ on $\mathcal{S}_0$ is constructed first (Section 5.1). This function serves us as an exact reference while constructing the functions at other levels. (b) C^1 data are computed for each face of each prism in each level (Section 5.3). (c) C^1 functions are constructed for each prism of any extracted scaffold that interpolates the vertices of the scaffold and fit $F^{(0)}$ by splines (Section 5.4).

5.1. Function over the finest level $\mathcal{S}_0$

The function $F^{(0)}$, whose level surfaces $F^{(0)}(x,y,z) = -1$ and $F^{(0)}(x,y,z) = 1$ will approximate the inner and outer surfaces, is constructed in two steps. First, function values and gradients (C^1 data) are defined on each of the faces of all the prisms, and second, the function is defined within the prisms, using the C^1 data on the prism faces.

Now we define C^1 data on the faces. Let $H_{lm}(t,\lambda)$ be a face of the prism P_{ijk} where $(l,m) \in \{(i,j),(j,k),(k,i)\}$. Then the function value on this face is defined by cubic Hermite interpolation on the line segment $[v_l(\lambda)\ v_m(\lambda)] = \{p \in \mathbb{R}^3\colon\ p = H_{lm}(t,\lambda), t \in [0,1]\}$ by interpolating the directional derivatives $D^s_{[v_m(\lambda)-v_l(\lambda)]^s}F(v_l(\lambda))$ and $D^s_{[v_m(\lambda)-v_l(\lambda)]^s}F(v_m(\lambda))$ for $s = 0,1$. Hence, $F(H_{lm}(t,\lambda))$ can be written as

$$\begin{aligned} F\bigl(H_{lm}(t,\lambda)\bigr) &= F\bigl(v_l(\lambda)\bigr)H_0^3(t) + F\bigl(v_m(\lambda)\bigr)H_2^3(t) \\ &\quad + \bigl[v_m(\lambda) - v_l(\lambda)\bigr]^{\mathrm{T}}\nabla F\bigl(v_l(\lambda)\bigr)H_1^3(t) \\ &\quad + \bigl[v_m(\lambda) - v_l(\lambda)\bigr]^{\mathrm{T}}\nabla F\bigl(v_m(\lambda)\bigr)H_3^3(t), \end{aligned} \tag{5.1}$$

where $H_0^3(t) = 1 - 3t^2 + 2t^3$, $H_1^3(t) = t - 2t^2 + t^3$, $H_2^3(t) = 3t^2 - 2t^3$, $H_3^3(t) = -t^2 + t^3$ are Hermite interpolation base functions, and

$$F\bigl(v_i(\lambda)\bigr) = 2\lambda - 1, \quad \nabla F\bigl(v_i(\lambda)\bigr) = (1-\lambda)N_i^{(0)} + \lambda N_i^{(1)}. \tag{5.2}$$

Here we have normalized the normals $N_i^{(0)}$ and $N_i^{(1)}$ such that $N_i^{\mathrm{T}} N_i^{(0)} = N_i^{\mathrm{T}} N_i^{(1)} = 2$ (recall that $N_i = V_i^{(1)} - V_i^{(0)}$), in order to have $D_{N_i} F = N_i^{\mathrm{T}} \nabla F$ on the edge $v_i(\lambda)$. Let

$$d_1(\lambda) = v_m(\lambda) - v_l(\lambda), \tag{5.3}$$

$$d_2(t) = (1-t)N_l + tN_m, \tag{5.4}$$

$$d_3(t,\lambda) = d_1 \times d_2. \tag{5.5}$$

Then we define the gradient $\nabla F(H_{lm}(t,\lambda))$ by the following conditions:

$$\begin{cases} d_1^{\mathrm{T}} \nabla F\big(H_{lm}(t,\lambda)\big) = \dfrac{\partial F(H_{lm}(t,\lambda))}{\partial t}, \\ d_2^{\mathrm{T}} \nabla F\big(H_{lm}(t,\lambda)\big) = \dfrac{\partial F(H_{lm}(t,\lambda))}{\partial \lambda}, \\ d_3^{\mathrm{T}} \nabla F\big(H_{lm}(t,\lambda)\big) = d_3^{\mathrm{T}} \nabla \breve{F}_{lm}(t,\lambda), \end{cases} \tag{5.6}$$

where

$$\nabla \breve{F}_{lm}(t,\lambda) = (1-t)\nabla F\big(v_l(\lambda)\big) + t\nabla F\big(v_m(\lambda)\big). \tag{5.7}$$

From (5.6) we have

$$\nabla F\big(H_{lm}(t,\lambda)\big)^{\mathrm{T}} = [P, Q, R][d_1, d_2, d_3]^{-1} \tag{5.8}$$

where

$$[d_1, d_2, d_3]^{-1} = \big[d_1\|d_2\|^2 - d_2\big(d_1^{\mathrm{T}} d_2\big), d_2\|d_1\|^2 - d_1\big(d_1^{\mathrm{T}} d_2\big), d_3\big]^{\mathrm{T}} / \|d_3\|^2,$$

and P, Q and R are the right-hand sides of (5.6).

Next we define C^1 functions within prisms. Let $[V_1 V_2 V_3]$ be a typical triangle pair. The C^1 function F in the prism P_{123} is defined by the side-vertex scheme defined by Theorem 3.1 in (Nielson, 1979):

$$F\big(p_{123}(b_1,b_2,b_3,\lambda)\big) = \sum_{i=1}^{3} w_i D_i(b_1,b_2,b_3,\lambda), \tag{5.9}$$

where $w_i = \prod_{j\neq i} b_j^2 / \sum_{k=1}^3 \prod_{j\neq k} b_j^2$, and D_i is defined by Hermite interpolation from the data on the prism faces (see (Bajaj and Xu, 1999) for details). Explicitly,

$$\begin{aligned} D_i(b_1,b_2,b_3,\lambda) \;=\; & F\big(p_i(b_1,b_2,b_3,\lambda)\big)H_0^3(b_i) \\ & + d_i(b_1,b_2,b_3,\lambda)^{\mathrm{T}} \nabla F\big(p_i(b_1,b_2,b_3,\lambda)\big)H_1^3(b_i) \\ & + F\big(v_i(\lambda)\big)H_2^3(b_i) + d_i(b_1,b_2,b_3,\lambda)^{\mathrm{T}} \nabla F\big(v_i(\lambda)\big)H_3^3(b_i), \end{aligned}$$

where

$$\begin{aligned} p_i(b_1,b_2,b_3,\lambda) &= \frac{b_i}{1-b_i} v_j(\lambda) + \frac{b_k}{1-b_k} v_k(\lambda), \\ d_i(b_1,b_2,b_3,\lambda) &= -\frac{b_j}{1-b_i} e_k(\lambda) - \frac{b_k}{1-b_i} e_j(\lambda), \end{aligned}$$

and $(i,j,k) \in \{(1,2,3),(2,3,1),(3,1,2)\}$, $e_k(\lambda) = v_j(\lambda) - v_i(\lambda)$, $e_j(\lambda) = v_k(\lambda) - v_i(\lambda)$.

5.2. *Minimal prism with ε offset*

As mentioned above, the shell $\{p \in \mathbb{R}^3\colon\ F^{(0)}(p) \in [-1, 1]\}$ constructed in this section is considered to be exact, and the other shells constructed later will approximate this shell to within the error ε. Therefore, this shell with its ε offset is required to be contained in all of the other scaffolds. This requirement will be one of the conditions in building the hierarchical representation of the scaffold. Since testing whether a triangular shell with ε offset is contained in another scaffold is time-consuming, we determine a minimal prism that contains the triangular shell with ε offset. In building the hierarchical representation, the shell containment requirement will be replaced by the minimal prisms containment requirement. Let $[V_i V_j V_k]$ be a triangle pair. Let $p^{(0)}$ be the point in $\mathbb{R}^3$ with P_{ijk}-coordinate $(b_1^{(0)}, b_2^{(0)}, b_3^{(0)}, \lambda^{(0)})$ and let $p^{(1)}$ be the intersection point of the surface $F^{(0)} = 1$ and the line $b_1^{(0)} v_i(\lambda) + b_2^{(0)} v_j(\lambda) + b_3^{(0)} v_k(\lambda)$, where they intersect at $\lambda = \lambda^{(1)}$. Then

$$\left\| p^{(0)} - p^{(1)} \right\| = \left| \lambda^{(0)} - \lambda^{(1)} \right| \left\| b_1^{(0)} N_i + b_2^{(0)} N_j + b_3^{(0)} N_k \right\|.$$

Then we require $|\lambda^{(0)} - \lambda^{(1)}| \geqslant \varepsilon/\sqrt{M}$, where $M := M(N_i, N_j, N_k)$ is the minimal value of the degree two Bézier polynomial $\|b_1 N_i + b_2 N_j + b_3 N_k\|^2$ on the triangle $\{b_1 + b_2 + b_3 = 1, b_i \geqslant 0\}$. Let $I_{ijk}^{\min} = [a, b]$ be the minimal interval such that $P_{ijk}(I_{ijk}^{\min})$ contains the triangular shell. Then we define the minimal prism as $P_{ijk}(I_{ijk}^{\varepsilon})$ with $I_{ijk}^{\varepsilon} = [a - \varepsilon/\sqrt{M}, b + \varepsilon/\sqrt{M}]$. The interval $[a, b]$ can be computed by numerical methods (see Section 2).

5.3. *Computation of face data*

The function F in each prism is defined by transfinite interpolation of the data on the face of the prism (see Section 5.1 or 5.4). To ensure that F is C^1 in the prism, the function and the gradient on the face need to be C^2 and C^1, respectively. Now we define the C^2 function $F(H_{lm})$ and C^1 gradient $\nabla F(H_{lm})$ on every face H_{lm} of each prism in every level. For the finest level, these functions have been defined by (5.1) and (5.6). Now we consider the functions on other levels. Though the face data on level $t + 1$ could be incrementally computed from the data of level t, we compute data on level $t + 1$ from level zero to avoid error accumulation. The DAG constructed enables us to trace back to $\mathcal{S}_0$ to locate the required data from level zero. Let

$$F\bigl(H_{lm}(t, \lambda)\bigr) = G_{lm}(t, \lambda) + \phi_{lm}^{\sigma}(t) + \psi_{lm}^{\sigma}(t)\lambda, \tag{5.10}$$

where $G_{lm}(t, \lambda)$ takes the same form as $F(H_{lm})$ in (5.1), and

$$\phi_{lm}^{\sigma}(t) = \sum_{i=2}^{2^\sigma - 2} \phi_i N_{i3}^{\sigma}(t), \quad \psi_{lm}^{\sigma}(t) = \sum_{i=2}^{2^\sigma - 2} \psi_i N_{i3}^{\sigma}(t),$$

where $\{N_{i3}^{\sigma}(t)\}_{i=-1}^{2^\sigma+1}$ are C^2 cubic B-spline basis functions defined on the uniform knots $t_i = i/2^\sigma$, $i = 0, 1, \ldots, 2^\sigma$. Here we shift N_{i3}^{σ} so that t_i is the center of the support $\operatorname{supp} N_{i3}^{\sigma} = ((i - 2)/2^\sigma, (i + 2)/2^\sigma)$. Note that the function values and the first order derivatives of ϕ_{lm}^{σ} and ψ_{lm}^{σ} are zero at the ends of the interval $[0, 1]$.

Since G_{lm} depends on vertex information only and it is easy to construct, we do not store the data of G_{lm}, but only ϕ_i and ψ_i. These parameters are determined by approximating the two intersection curves of the finest level surfaces $F^{(0)} = \pm 1$ with the face H_{lm}, in the least square sense:

$$\int_0^1 \bigl[F\bigl(H_{lm}(t, \lambda_s(t))\bigr) + (-1)^s\bigr]^2 \, \mathrm{d}t = \min, \quad s = 0, 1, \tag{5.11}$$

where $\lambda_s(t)$, for fixed t, is defined by the intersection point of the line $(1-t)v_l(\lambda) + t v_m(\lambda)$ with the surface $F^{(0)} + (-1)^s = 0$. The required pieces of the intersection are obtained from the DAG. The minimization in (5.11) leads to a system of linear equations

$$\sum_{i=2}^{2^\sigma - 2} \int_0^1 \bigl(\phi_i + \psi_i \lambda_s(t)\bigr) N_{i3}^\sigma(t) N_{j3}^\sigma(t) \, \mathrm{d}t = c_j$$

with $c_j = -\int_0^1 [G_{lm}(t, \lambda_s(t)) + (-1)^s] N_{j3}^\sigma(t)\, \mathrm{d}t$ and $j = 2, \ldots, 2^\sigma - 2,\ s = 0, 1$. The integrations in the system are computed by Gauss–Legendre quadrature rule on each of the sub-intervals $[i/2^\sigma, (i+1)/2^\sigma]$ and then summed up. This order $2(2^\sigma - 2)$ equation can be solved by solving two order $2^\sigma - 2$ linear systems. The intersection point $\lambda_s(t)$ is computed by Newton iteration, and the integer σ is chosen on trial bases. Starting from $\sigma = 1$, we solve the equation and compute the least square error. If the error is larger than the given ε, then increase σ by one, until the error is within the tolerance.

Next, we define the gradient $\nabla F(H_{lm}(t, \lambda))$ by the conditions (5.6), but $\breve{F}_{lm}(t, \lambda)$ is modified by adding a spline function:

$$\nabla \breve{F}_{lm}(t, \lambda) = (1-t)\nabla F\bigl(v_l(\lambda)\bigr) + t \nabla F\bigl(v_m(\lambda)\bigr) + \breve{\phi}_{lm}^\sigma(t) + \breve{\psi}_{lm}^\sigma(t)\lambda \tag{5.12}$$

with

$$\breve{\phi}_{lm}^\sigma(t) = \sum_{i=2}^{2^\sigma - 2} \breve{\phi}_i N_{i3}^\sigma(t), \qquad \breve{\psi}_{lm}^\sigma(t) = \sum_{i=2}^{2^\sigma - 2} \breve{\psi}_i N_{i3}^\sigma(t),$$

where $\breve{\phi}_i, \breve{\psi}_i \in \mathbb{R}^3$ are determined by

$$\int_0^1 \bigl\| \nabla \breve{F}_{lm}\bigl(t, \lambda_s(t)\bigr) - \nabla F^{(0)}\bigl(H_{lm}(t, \lambda_s(t))\bigr) \bigr\|^2 \, \mathrm{d}t = \min \tag{5.13}$$

for $s = 0, 1$, and $\lambda_s(t)$ is defined as before. (5.13) can be solved together with (5.11) since they share the same coefficient matrix.

5.4. Construction of C^1 spline approximations

In this section, we construct a piecewise C^1 function $F = F_\sigma$ ($\sigma \geqslant 0$ fixed) over the collection of the volumes, such that it (Hermite) interpolates the C^1 data and fits $F^{(0)}$. To achieve ε approximation and multiresolution representation in the h-direction, spline functions defined on triangles are utilized in the construction of F. On a triangular domain

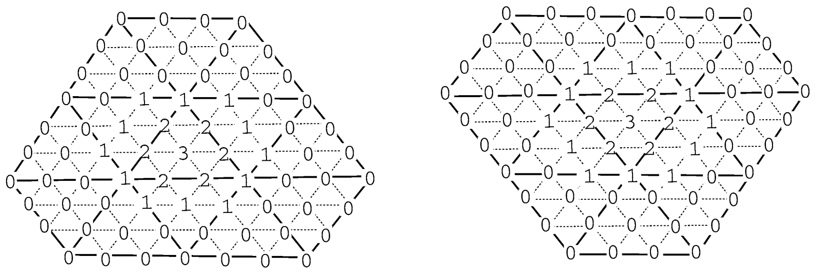


Fig. 5.1. Bézier coefficients for two C^1 cubic spline basis functions. Each is defined on the union of 13 sub-triangles, which forms the support of the function.

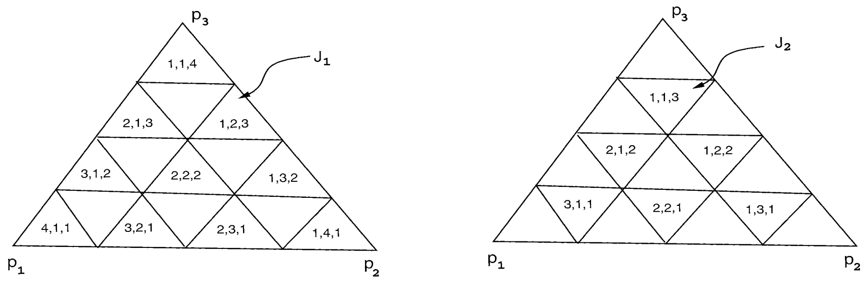


Fig. 5.2. For the regular partition of a triangle with resolution 2^σ, the index set J^σ of the sub-triangles is divided into J_1^σ and J_2^σ. This figure shows them for $\sigma = 2$.

with a regular partition, C^1 cubic splines defined in BB form were given by Sabin (1976). Fig. 5.1 gives the BB-form coefficients of a typical base function defined on 13 sub-triangles. Note that these splines in general are not linearly independent (see (Böhm et al., 1984)). However, the collection we use is indeed linearly independent. For a regular partition of a triangle T, we shall associate a base function to each sub-triangle of the partition. To give proper indices for these bases, we label the sub-triangles as T_{ijk} for $(i,j,k) \in J^\sigma = J_1^\sigma \cup J_2^\sigma$, where J_1^σ and J_2^σ are defined as follows:

$$J_1^\sigma = \bigl\{(i,j,k)\colon\ i,j,k \in \{1,2,3,\ldots,2^\sigma\};\ i+j+k = 2^\sigma + 2\bigr\},$$
$$J_2^\sigma = \bigl\{(i,j,k)\colon\ i,j,k \in \bigl\{1,2,3,\ldots,2^\sigma - 1\bigr\};\ i+j+k = 2^\sigma + 1\bigr\},$$

where 2^σ is the resolution of the partition. Fig. 5.2 gives J_1 and J_2 for $\sigma = 2$. Now we denote the base function defined by Fig. 5.1 with center triangle T_{ijk} as N_{ijk}^σ.

5.4.1. F on prisms

Let $[V_1V_2V_3]$ be a typical triangle pair. Define

$$F_\sigma\bigl(p_{123}(b_1,b_2,b_3,\lambda)\bigr) = \sum_{i=1}^{3} w_i D_i(b_1,b_2,b_3,\lambda) + T_\sigma(b_1,b_2,b_3,\lambda), \tag{5.14}$$

where the first term of left-hand side is in the same form as (5.9), and the second term is a spline function:

$$T_\sigma(b_1,b_2,b_3,\lambda)=\sum_{(i,j,k)\in J_3^\sigma}(a_{ijk}+w_{ijk}\lambda)N_{ijk}^\sigma(b_1,b_2,b_3)$$

with $J_3^\sigma=\{(i,j,k)\in J^\sigma\colon\ i>1, j>1, k>1\}$. This is called a *correction term*, which is used to fit the finest level shell surface in the least square sense:

$$\iint_\Delta\bigl[F_\sigma\bigl(b_1,b_2,b_3,\lambda_s(b_1,b_2,b_3)\bigr)-(-1)^s\bigr]^2\,\mathrm{d}S=\min \tag{5.15}$$

for $s=0,1$, where $\lambda_s(b_1,b_2,b_3)$ for each (b_1,b_2,b_3) is defined by the intersection point of the line $b_1v_1(\lambda)+b_2v_2(\lambda)+b_3v_3(\lambda)$ with the surface $F^{(0)}+(-1)^s=0$. The required pieces of the intersection are obtained from the DAG. The domain Δ in the integration is the unit triangle defined by $\{(b_1,b_2,b_3)\colon\ b_1+b_2+b_3=1, b_i\geqslant 0\}$. The minimization in (5.15) leads to a system of linear equations.

5.4.2. Hierarchical representation of correction term

In the construction of $F=F_\sigma$, we have associated it with an integer σ. This integer indicates the level of the hierarchical multiresolution representation of F in the h-direction. However, the construction and expression of F in Section 5.4.1 is not incremental, as the construction of $F_{\sigma+1}$ does not utilize the information of F_σ. In this subsection, we revise some parts of the construction in Section 5.4.1, so that F is progressively constructed. Now we want to have the following form expression:

$$T_\sigma=T_{\sigma-1}+\sum_{(i,j,k)\in J_3^\sigma\backslash 2*J_3^{\sigma-1}}\bigl(a_{ijk}^\sigma+w_{ijk}^\sigma\lambda\bigr)N_{ijk}^\sigma,$$

where $T_1=0$. Let

$$W^\tau=\mathrm{span}\bigl\{N_{ijk}^\tau\colon\ i\in J_3^\tau\setminus 2*J_3^{\tau-1}\bigr\},\qquad S^\tau=W^2\oplus W^3\oplus\cdots\oplus W^\tau.$$

Then S^τ is a C^1 cubic spline function space on a triangle partitioned regularly with resolution 2^τ. Once $T_{\sigma-1}$ has been defined, the coefficients a_{ijk}^σ and w_{ijk}^σ are computed by fitting $F^{(0)}$ in the volume. Since the elements in S^τ have zero function value and zero first order partial derivative values on the boundary of the triangle, we can use different σ for different prisms to get an adaptive construction without destroying the continuity of the composite function. For the prism P_{ijk}, let ε_{ijk}^σ be the fitting error. Then for any given fitting error tolerance ε, we can choose a minimal σ so that $\varepsilon_{ijk}^\sigma\leqslant\varepsilon$. This σ is prism dependent.

Basic result. *The composite function F, defined on any extracted scaffold and for any varying and prism-dependent $\sigma\geqslant 0$, is C^1.*

We omit the tedious mathematical derivation of the proof of this result, but do give examples illustrating the smoothness of the function F_σ. Fig. 5.3 provides examples of hierarchical multiresolution construction in the h-direction. The figures (for $\sigma=1,2,3$)

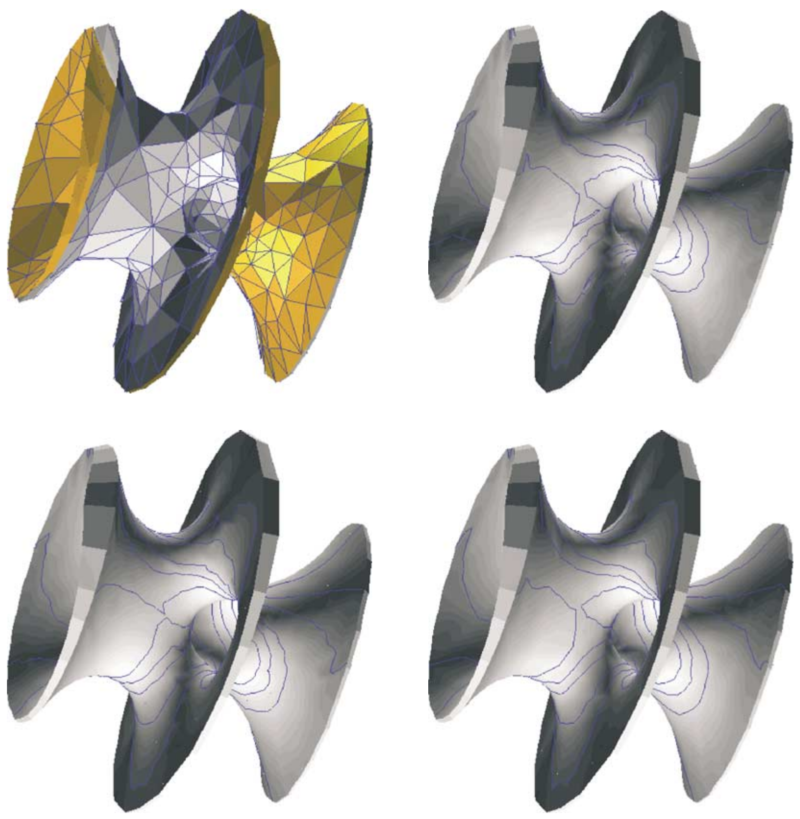


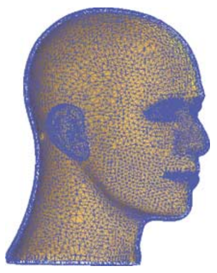
Fig. 5.3. h-direction hierarchical multiresolution construction for the hypersheet surface triangulation (513 triangle pairs) of level 2 of the DAG with $\sigma = 1$ (top-right), $\sigma = 2$ (bottom-left) and $\sigma = 3$ (bottom-right). The level 0 triangulation has 917 triangle pairs with varying thickness.

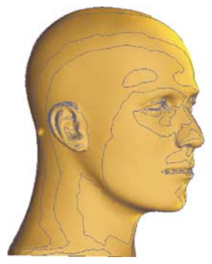
with continuous isophotes show the constructed surface is indeed smooth. The figures also show the surface shape improvement by the splines when σ is increased.

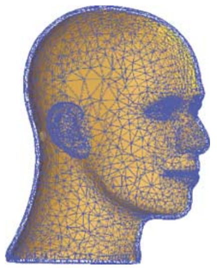
The H-direction multiresolution is shown in Fig. 5.4. To see the pairwise nature of the triangulation, $\mathcal{T}^{(1)}$ is plotted by wire, while $\mathcal{T}^{(0)}$ is plotted by plane segments. Fig. 5.4(a) is the original matched triangulation pair that has 25561 triangle pairs. Fig. 5.4(b), (c) and (d), which have 13141, 6765, and 4069 triangle pairs respectively, are the extracted triangulations. More examples are given in Fig. 8.1.

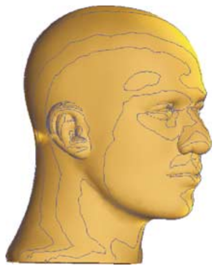
6. Capturing curve creases

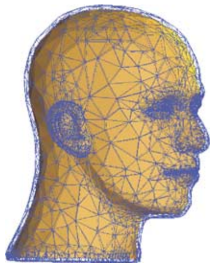
To capture sharp curve creases, we need to mark certain edges as sharp. To this end, we compute the dihedral angle $\theta = \pi - \theta_1$ for the two incident faces, for each edge of the triangulations $T^{(0)}$ and $T^{(1)}$. If $\theta < \alpha$, then this edge is marked as a sharp edge. Here θ_1 is the angle between the normals of the two triangles and α is a threshold value for

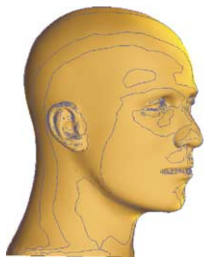

(a) 25552 triangle pairs, $g = 0°$

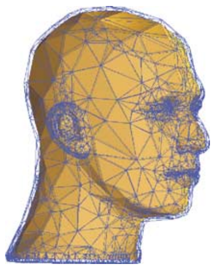

(a1) Inner boundary


(b) 13128 triangle pairs, $g = 5°$


(b1) Outer boundary


(c) 6754 triangle pairs, $g = 15°$


(c1) Inner boundary


(d) 4074 triangle pairs, $g = 30°$

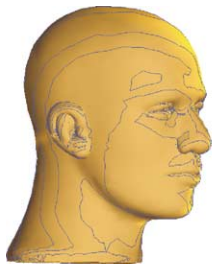

(d1) Outer boundary

Fig. 5.4. H-direction smooth reconstruction of matched triangulation pairs: (a) the input triangulation. (b), (c) and (d) are adaptively extracted meshes from the DAG with $g = 5°, 15°, 30°$, respectively. The right column shows inner/outer boundary surfaces with isophotes showing the smoothness. The level in the h-direction is zero.

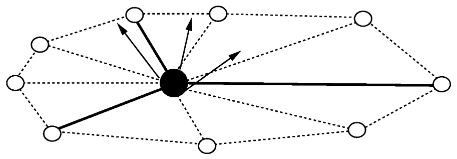


Fig. 6.1. Grouping the triangles by the sharp edges (thick lines) and assigning a normal for each group.

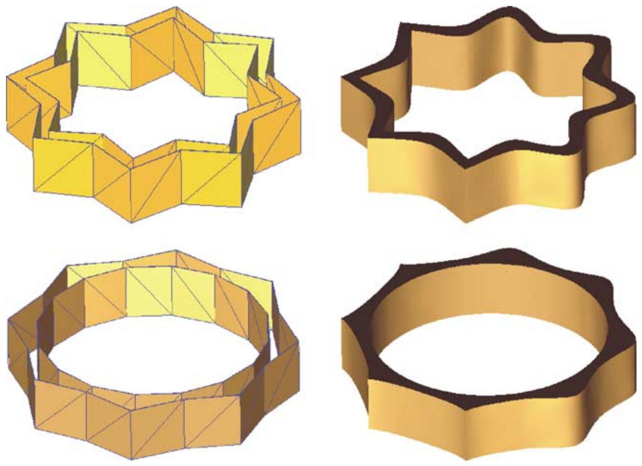


Fig. 6.2. Left: the input polygons with some edges marked as sharp. Right: the constructed shell surfaces with sharp curve creases. There are four edge pairs (inner and outer) on the top polygon marked as sharp. On the bottom polygon, only four outer edges are marked.

controlling the sharp curve crease. After marking the edges, the vertices also need to be marked. If there exist sharp edges incident to a vertex, then we say this vertex is sharp, otherwise, it is non-sharp. For a sharp vertex, the normal that has been assigned before needs to be re-computed. The triangles around a sharp vertex are divided into some groups by the sharp edges (see Fig. 6.1). For each group, we assign a single normal for the vertex. This normal may be computed as the weighted average of the face normals. The weight is chosen to be the angle of the edges that are incident to this vertex. In the construction of the surface patch for one triangle, there is only one normal used for each vertex of the triangle. This normal is the vertex normal if the vertex is non-sharp, otherwise the normal is the group's normal.

For a sharp edge $[V_l V_m]$, the function $F(H_{lm})$ defined in (5.10) and the gradient $\nabla \breve{F}_{lm}$ defined in (5.12) need to be redefined. The function $F(H_{lm}(t,\lambda))$ is changed to be the average function of the two local fitting functions that are defined on the neighbor volumes of the face. If there is only one neighbor volume, as is the case for a boundary edge, the face function is taken to be the volume function on that face. The average function makes the composite function continuous on the face H_{lm}. For the gradient on the face, we need to define two gradient functions, each being from one of the two volume functions. That is, the gradient function $\nabla \breve{F}_{lm}$ in (5.13) is replaced by the gradient of the volume functions on this face. Hence, the gradient of the composite function is not continuous at the face. Nevertheless, this gradient interpolates the sharp normals on the two vertices. Hence, sharp curve creases are captured.

Two examples are shown in Fig. 6.2. The left two figures are input polygons, and the right two figures are the shell bodies that are the corresponding output. In the star-like polygon on the top-left, the left four inner and outer peak edges are labeled as sharp. The shell surface on the top-right exhibits the sharp curve creases. For the bottom-left polygon, the left four peak edges of the outer polygon are labeled as sharp, and no edge is labeled as sharp for the inner polygon. The figure on the bottom-right shows the outer sharp, inner smooth nature. Another example that has sharp curve creases is shown in Fig. 8.1(f) and (f1).

7. Evaluation and display of shell surfaces

Often we wish to evaluate the surface $F = \alpha$ for a given $\alpha \in [-1, 1]$. Let $[V_i V_j V_k]$ be any triangle pair. Then for each (b_1, b_2, b_3), $b_i \geqslant 0$, $\sum b_i = 1$, determine

$$\lambda_{\min}^{(\alpha)} = \lambda_{\min}^{(\alpha)}(b_1, b_2, b_3)$$

such that

$$\begin{aligned} &F\bigl(p_{ijk}(b_1, b_2, b_3, \lambda_{\min}^{(\alpha)})\bigr) = \alpha, \\ &\bigl|\lambda_{\min}^{(\alpha)} - \tfrac{1}{2}\bigr| = \min\bigl\{\bigl|\lambda - \tfrac{1}{2}\bigr|\colon\ F\bigl(p_{ijk}(b_1, b_2, b_3, \lambda)\bigr) = \alpha\bigr\}. \end{aligned} \tag{7.1}$$

The surface point is defined by

$$p = p_{ijk}(b_1, b_2, b_3, \lambda_{\min}^{(\alpha)}).$$

The main task here is to compute $\lambda_{\min}^{(\alpha)}$ for each (b_1, b_2, b_3). It follows from (5.9) that $D_i(b_1, b_2, b_3, \lambda)$ is a rational function of λ. It is in the form

$$f_0 + f_1\lambda + f_2\lambda^2 + \frac{N_0 + N_1\lambda + N_2\lambda^2 + N_3\lambda^3 + N_4\lambda^4}{D_0 + D_1\lambda + D_2\lambda^2}. \tag{7.2}$$

Hence $\phi(\lambda) := F(p_{ijk}(b_1, b_2, b_3, \lambda))$ is a rational function of λ of the form (7.2). The nearest zero to 1/2 of $\phi(\lambda) - \alpha$ is the required $\lambda_{\min}^{(\alpha)}$. Although $\phi(\lambda) - \alpha = 0$ is a nonlinear algebraic equation, $\phi(\lambda) - \alpha$ can be approximated by a polynomial of degree at most 2, since the rational term in (7.2) is small compared with the polynomial term. Hence, taking the roots of the polynomial part as an initial value and then using Newton iteration, we obtain the required solution. The computation shows that this approach is very effective.

In addition to extracting the boundary surfaces of a shell, we can also extract the surfaces between the boundaries by taking $\alpha \in (-1, 1)$. Furthermore, by taking a sequence of α in increasing order, the shell can be divided into layers (see Fig. 7.2), an onion-like structure.

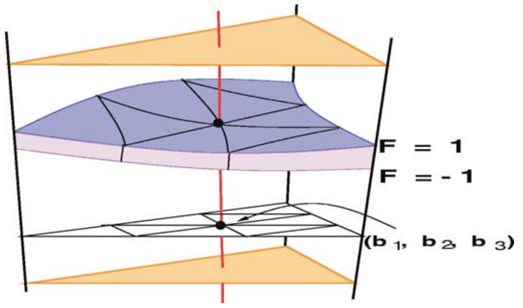


Fig. 7.1. Evaluation of a shell surface: The surface is parameterized in each prism with the triangle $\{b_1 + b_2 + b_3 = 1,\ b_i \geqslant 0\}$ as its domain.

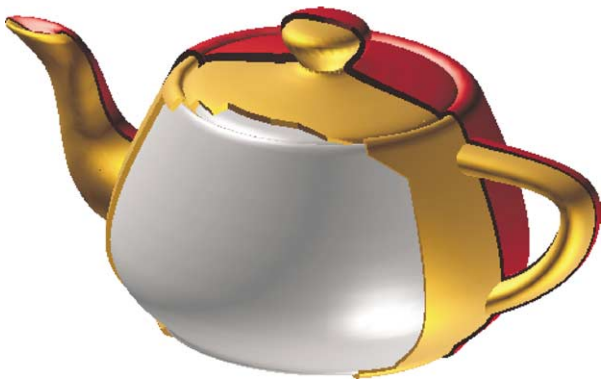
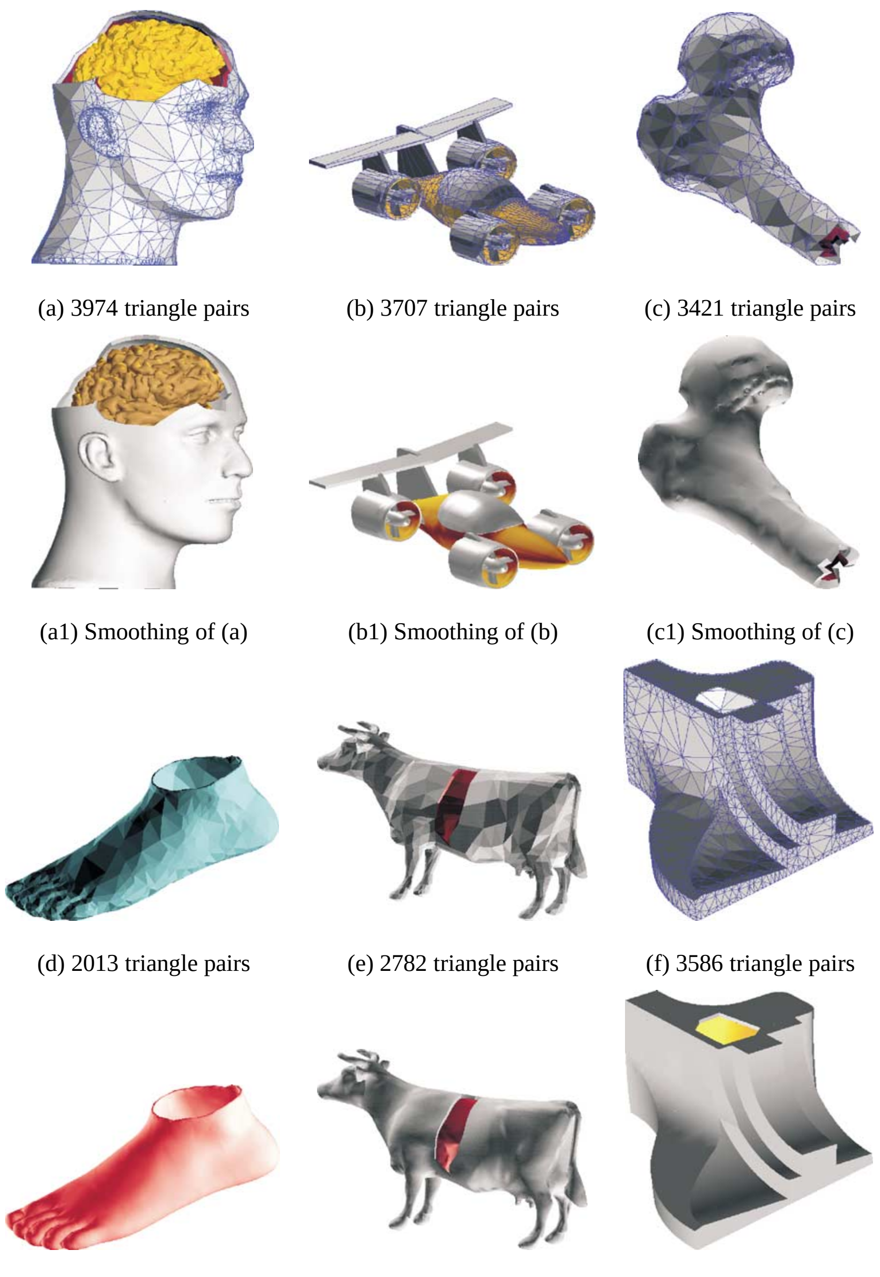


Fig. 7.2. Middle surface and multiple layers of a shell: Inner surface and two layers are presented.

Using the hierarchical representation of the correction term, the evaluation of the shell surface can be progressive in the h-direction. Since $F_\sigma - F_{\sigma+1}$ in general is small, $F_\sigma = \alpha$ is a good approximation of $F_{\sigma+1} = \alpha$. Hence in the Newton iteration for getting the surface point on $F_{\sigma+1} = \alpha$, $F_\sigma = \alpha$ is a very good initial value. Furthermore, coefficients in (7.2) are the same for F_σ and $F_{\sigma+1}$, except for f_0 and f_1 which are affected by the correction term. H-direction progressive evaluation (from coarse to fine) is also possible in theory but it is not as cheap as the h-direction, since an extracted scaffold may combine different levels. Hence, we do not recommend the H-direction progressive evaluation.

8. Conclusions

We have presented an adaptive hierarchical Hh-multiresolution reconstruction algorithm to model smooth shell surface objects from a matched triangulation pair. The implementation and test (see Fig. 8.1 for more examples) show that the proposed method is correct and fulfills our purpose.



(a) 3974 triangle pairs (b) 3707 triangle pairs (c) 3421 triangle pairs

(a1) Smoothing of (a) (b1) Smoothing of (b) (c1) Smoothing of (c)

(d) 2013 triangle pairs (e) 2782 triangle pairs (f) 3586 triangle pairs

(d1) Smoothing of (d) (e1) Smoothing of (e) (f1) Smoothing of (f)

Fig. 8.1. Shell surface constructions: (a)–(f) are the extracted triangulations from the DAGs with $g = 30^\circ, 5^\circ, 40^\circ, 30^\circ, 30^\circ$ and 30°, respectively. (a1)–(f1) are the corresponding shell surface reconstruction with error $\varepsilon = 0.05$. The original triangulations have 25552 (head), 4629 (aircar), 7798 (femur), 20216 (foot), 5804 (cow), and 12946 (fandisk) triangle pairs, respectively.

Appendix A. Vertex removal

Consider the vertex removal of the matched triangulation pair $\mathcal{T}_t$. This procedure is subject to the following conditions:

Containment. The initial minimal prism scaffold is always contained in the new scaffold when a vertex is removed and the hole left is re-triangulated (see Section 5.2 for computing the minimal prism scaffold).

Lower bound of angle. The angles of triangles resulting from the removal and re-triangulation are not smaller than the given control angle θ. Enforcing this condition avoids producing thin triangles.

Under these conditions, we first subdivide all the vertex pairs of $\mathcal{T}_t$ into two groups. One group contains the vertices that are not removable, and the other contains the remaining vertices. The unremovable group includes vertices that are marked as sharp (see Section 6), and those that if they are removed, the resulting prisms do not satisfy the containment condition or the resulting triangles do not satisfy the lower bound condition.

There is plenty of literature on polygonal mesh simplification (see, e.g., (Erikson, 1996; Gieng et al., 1998; Guskov et al., 1999; Hamann, 1993, 1994; Hoppe et al., 1994; Lee et al., 1998; Schroeder et al., 1992). Any vertex removal scheme, for instance the scheme created by Hamann (1993, 1994), that is based on the local curvature estimation can be adjusted to serve our purpose. To have the adaptiveness property and to utilize the normal information provided, our vertex removal criterion is based on normal variation. The flattest part of the triangulation is removed first. Let R_t be the set of removable vertices of $\mathcal{T}_t$. Then for each vertex $v \in R_t$, we specify a grade to it that measures the normal variation. After all the vertices in R_t are graded, we sort the grades in increasing order. Then start the removal from the vertex that has lowest grade. Of course, when one vertex is removed, its neighbor vertices become unremovable.

In order to define the grade of a point, we introduce sets of points G_{jkl}. These sets will consist of removed points that lie in the prism P_{jkl}. At level 0, these sets are empty, which is equivalent to saying that $G_{ijk} = \emptyset$ for each T_{ijk} in the original triangulation $\mathcal{T}_0$. At level 1, each of these sets can contain at most two points: if T_{jkl} is one of the resulting triangles formed by the removal of the point p, then G_{jkl} can only contain p and its corresponding point in the other layer (inner or outer). The point sets G's for each higher level are defined recursively as follows. For any $p_i \in R_t$, we first define $K_i = \{(i, j, k)\colon\ T_{ijk} \in \mathcal{T}_t^{(0)}\}$ to be the set of all triples of indices corresponding to triangles in the triangulation $\mathcal{T}_t^{(0)}$ around p_i, or the corresponding point in the inner layer if p_i is in the outer layer. Then let K_i' be the set of all triples of indices (j, k, l), taken from the indices of points connected to p_i, corresponding to the triangles formed by the re-triangulation after p_i is removed. Finally, define G_{jkl} as the regrouping of the data set $p_i \cup (\bigcup_{(i,j,k)\in K_i} G_{ijk})$ into the prisms $P_{jkl},\ (j, k, l) \in K_i'$. Therefore, the sets G_{jkl} at level $t + 1$ are recursively generated, with each being the regrouping of the union of similar sets the level t with a newly removed vertex and its corresponding vertex in the other layer.

Now for each vertex p in G_{jkl} we consider four angles that measure the normal variation. For $s = 0, 1$, we define $\theta^{(s)}_{jkl}(p)$ to be the angle between the averaging normal $b^{(s)}_1 N^{(s)}_j + b^{(s)}_2 N^{(s)}_k + b^{(s)}_3 N^{(s)}_l$ and the normal N of vertex p, and $\tilde{\theta}^{(s)}_{jkl}(p)$ to be the angle between the normal $N^{(s)}_{jkl}$ of the triangle $T^{(s)}_{jkl}$ and normal N. (N is either $N^{(0)}$ or $N^{(1)}$, depending on whether p is in the inner or outer layer.) The sub-grade of p with respect to (j, k, l) is the largest of these four angles, with some possibly multiplied by a weight. Then the grade of a point $p_i \in R_t$ is defined to be the maximum of the sub-grades of all points p in any of the prisms resulting from the re-triangulation occurring upon the removal of p_i. Specifically, we define

$$\text{Sub-grade}(p;\, j, k, l) = \max_{s=0,1} \max\bigl\{\theta^{(s)}_{jkl}(p),\, w\tilde{\theta}^{(s)}_{jkl}(p)\bigr\},$$

and

$$\text{Grade}(p_i) = \max_{(j,k,l)\in K'_i} \max_{p\in G_{jkl}} \bigl[\text{Sub-grade}(p;\, j, k, l)\bigr], \tag{A.1}$$

where w is a weight whose value we choose to equal 2. This grade is assigned to each of the prisms yielded from the re-triangulation for the later use of scaffold extraction. The initial prisms in $\mathcal{S}_0$ are assigned zero grade. We use the same notation Grade$(\cdot)$ to denote the grade of a prism.

Appendix B. Re-triangulation

After p_i is removed, the resulting hole is re-triangulated. We only consider the re-triangulation of one of the triangular surfaces that make up the matched triangulation pair. The other surface is triangulated in the same manner, preserving similar topology for the twin. Let $p^{(1)}_0, p^{(1)}_1, \ldots, p^{(1)}_n$ be the vertex chain on the positive side that produces the hole. We assume the chain is arranged in a counterclockwise manner. The hole is triangulated by:

(1) If $n = 2$, one triangle is returned.

(2) If $n \geqslant 3$, then label the point $p^{(1)}_j$ as convex if (see Fig. B.1) $[(p^{(1)}_{j+1} - p^{(1)}_j) \times (p^{(1)}_{j-1} - p^{(1)}_j)]^{\mathrm{T}} n^{(1)}_j > 0$, and label it as non-convex otherwise.

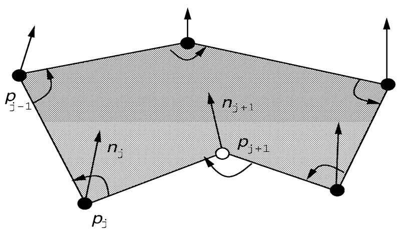


Fig. B.1. Convexity test by right-hand rule: The vertices with black and white dots are labeled as convex and non-convex, respectively.

(3) Find (j, k), $0 \leqslant j, k \leqslant n$, such that
 (i) $p_j^{(1)}$ and $p_k^{(1)}$ are not adjacent in the closed chain $\{p_i^{(1)}\}$.
 (ii) $p_j^{(1)}$ and $p_k^{(1)}$ are selected from the non-convex vertices when there are two or more non-adjacent points labeled as non-convex. If there is only one non-convex point, or exactly two non-convex points that are adjacent, then one of $p_j^{(1)}$ and $p_k^{(1)}$ must be non-convex.
 (iii) Under the conditions (i) and (ii), the geodesic distance from $p_j^{(1)}$ to $p_k^{(1)}$ on the previous (i.e., level t) triangulation $\{T_{ijk}\}$, $(i, j, k) \in K_i$ is minimal.

Next divide the points surrounding the hole into two chains $[p_j^{(1)}, p_{j+1}^{(1)}, \ldots, p_k^{(1)}]$ and $[p_k^{(1)}, p_{k+1}^{(1)}, \ldots, p_j^{(1)}]$, where each chain is again arranged in counterclockwise order and the indices are considered modulo n. For each hole, repeat steps (1)–(3).

References

Bajaj, C., Xu, G., 1999. Smooth multiresolution reconstruction of free-form fat surfaces, TICAM Report 99-08. Texas Institute for Computational and Applied Mathematics, The University of Texas at Austin.

Bernadou, M., Boisserie, J.M., 1982. The Finite Element Method in Thin Shell Theory: Application to Arch Dam Simulations. Birkhäuser.

Böhm, W., Farin, G., Kahmann, J., 1984. A survey of curve and surface methods in CAGD. Computer Aided Geometric Design 1, 1–60.

Bucalem, M.L., Bathe, K.J., 1997. Finite element analysis of shell structures. Archives Comput. Methods Engrg. 4, 3–61.

Cirak, F., Ortiz, M., Schroder, P., 1999. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis, Technical Report, http://www.multires.caltech.edu/pubs/.

de Berg, M., Dobrindt, K.T.G., 1998. On levels of detail in terrains. Graphical Models and Image Processing 60, 1–12.

De Floriani, L., Magillo, P., Puppo, E., 1999. Data structures for simplicial multi-complexes, in: Güting, R.H., Papadias, D., Lochovsky, F. (Eds.), Proc. 6th International Symposium on Spatial Databases, Hong Kong.

Erikson, C., 1996. Polygonal simplification: An overview, Technical Report TR96-016. University of North Carolina, Chapel Hill.

Farin, G., 1990. Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide. Academic Press.

Gieng, T.S., Hamann, B., Joy, K.I., Schussman, G.L., Trotts, I.J., 1998. Constructing hierarchies for triangle meshes. IEEE Transactions on Visualization and Computer Graphics 4 (2), 145–161.

Guskov, I., Sweldens, W., Schröder, P., 1999. Multiresolution signal processing for meshes, in: Proceedings of SIGGRAPH-99, pp. 325–334.

Hamann, B., 1993. Curvature approximation for triangulated surfaces, in: Farin, G., Hagen, H., Noltemeier, H. (Eds.), Geometric Modelling, Computing Supplement 8. Springer, Berlin, pp. 139–153.

Hamann, B., 1994. A data reduction scheme for triangulated surfaces. Computer Aided Geometric Design 11 (2), 197–214.

Hoppe, H., DeRose, T., Duchamp, T., Halstend, M., Jin, H., McDonald, J., Schweitzer, J., Stuetzle, W., 1994. Piecewise smooth surfaces reconstruction, in: Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH-94, pp. 295–302.

Hoschek, J., Lasser, D., 1993. Fundamentals of Computer Aided Geometric Design. A.K. Peters.

Ketchum, M.S., Ketchum, M.A., 1997. Types and forms of shell structures, http://www.ketchum.org/ShellTandF/index.html.

Lee, A.W.F., Sweldens, W., Schröder, P., Cowsar, L., Dobkin, D., 1998. MAPS: Multiresolution adaptive parameterization of surfaces, in: Proceedings of SIGGRAPH-98, pp. 95–104.

Liu, W.K., Guo, Y., Tang, S., Belytschko, T., 1998. A multiple-quadrature eight-node hexahedral finite element for large deformation elastoplastic analysis. Computer Methods in Applied Mechanics and Engineering 154, 69–132.

Mollmann, H., 1981. Introduction to the Theory of Thin Shells. Chichester, New York.

Nielson, G.M., 1979. The side-vertex method for interpolation in triangles. J. Approx. Theory 25, 318–336.

Piegl, L., Tiller, W., 1997. The NURB Book. Springer, Berlin.

Sabin, M., 1976. The use of piecewise form of numerical representation of shape, Ph.D. Thesis. Hungarian Academy of Science, Budapest.

Schroeder, G., Zarge, J.A., Lorensen, W.E., 1992. Decimation of triangle meshes, in: Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH-92, pp. 65–70.

Szabo, B., Babuska, I., 1991. Finite Element Analysis. Wiley, New York.