Hierarchical Multiresolution Reconstruction of Shell Surfaces

Chandrajit L. Bajaj
Department of Computer Science,
University of Texas, Austin, TX 78712

Guoliang Xu
State Key Laboratory of Scientific and Engineering
Computing, Chinese Academy of Sciences, Beijing

Robert J. Holt Arun N. Netravali
Bell Laboratories, Lucent Technologies
Murray Hill, NJ 07974

March 1, 2000

Abstract

Several naturally occurring as well as manufactured objects have shell-like structures, that is, their boundaries consist of surfaces with thickness. In this paper we call these shell structures "fat surfaces". We present an adaptive, hierarchical Hh-multiresolution reconstruction algorithm to model fat surface objects from a matched triangulation pair. Fat surfaces are constructed by the contours of trivariate functions defined on prism scaffolds. 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 fat surfaces can be made to capture sharp curve creases on the shell while being C^1 smooth everywhere else. We also allow function values to be attached to the input vertices of the shell triangulations, so that physical data fields defined on the shell can be visualized and texture mapping can be performed. Using an interval of isocontours 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.

Key words. Curve & Surfaces, Geometric Modeling, Level of Detail Algorithms, Mesh Generation

1 Introduction

Many human manufactured and several naturally occurring objects have shell-like structures, that is, the object bodies consist of surfaces with thickness. We call such surfaces *fat surfaces*. Our use of the term fat surfaces differs from that of Barnhill, Opitz and Pottmann [5], where this term is used to denote functions on surfaces. The problem of constructing smooth approximations to fat surface objects 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 and Background

In engineering, shell structures are often analyzed by finite element methods (see [6, 8, 10, 27, 33]). 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 [26]) or pentahedral (see [33]). In these cases, the output could be a matched triangulation pair. In this paper, our aim is to reconstruct smooth shells from triangulation. However, we do not assume that the shell is uniform in thickness; instead, we are given two triangulations which represent two boundaries of the shell. These triangulations could 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 often must 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 $\mathcal{T} = \{\mathcal{T}^{(0)}, \mathcal{T}^{(1)}\}$ (also called a fat triangulation), with attached normals at each vertex and possibly a vector of function values, which presents a linearization of the inner and outer boundary surfaces of a shell domain, also, we are given a error control tolerance $\epsilon > 0$. The goal is to reconstruct hierarchical multiresolution smooth fat surfaces whose bounding surfaces provide approximations of $\mathcal{T}^{(0)}$ and $\mathcal{T}^{(1)}$, re-*

spectively, with errors no larger than ϵ. Furthermore, if function values are attached, smooth functions on the shell are also constructed that interpolate the function values.

The purpose of allowing the attaching of vertex function values in our problem is to model physical data, such as stress, temperature, and density etc., or graphics data, such as (r, g, b) or the data for texture mapping.

In contrast with the *geometric modeling* problem of shell construction, we refer to the problem of function construction on the shell as *functional modeling*. In this paper, the functions on the shell are constructed in the same pipeline as the shell with the different "feed" to the pipeline. Hence our attention is focused on the geometric problem, but still addresses the difference of the functional one.

The hierarchical scheme is comprised of multiresolutions in two directions. The H-direction multiresolution is a level-of-detail (LOD) representation of the (irregular) fat triangular mesh (we use "H" to represent it). The h-direction multiresolution is the regular subdivision of each of the fat triangles (we use "h" to represent it). The terminology Hh-*multiresolution* is an imitation of hp-*version* in finite element analysis (see [33]), where h-version means mesh size h decrease and p-version means the degree of shape functions (polynomial) increase. Our H-direction corresponds to the h-version. However, 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.

Apart from finite element analysis, another source for producing matched triangulation pairs is the set of nearby iso-contours of volume data. For example, if one iso-surface triangulation represents the boundary of a shell structure in a volume, for instance the bone in medical CT data, the other boundary triangulation could be obtained by shooting the surface with the straight rays starting from the vertices and proceeding in the normal direction (see Fig. 1.1). To obtain a matched triangulation pair from two general point clouds, we use local data fitting and project each one onto the other (see Appendix for detail) to first pair up the vertices, and then triangulate one cloud of points. The other is correspondingly triangulated to get a matched triangulation pair. Algorithms for triangulating surface data may be found in [2, 3, 4, 12, 14, 23, 9, 28]. We use the scheme of [2]. The Cohen et al.'s envelope approach for polygonal model simplification [11] can produce LOD matched triangulation pairs that can be used as our input.

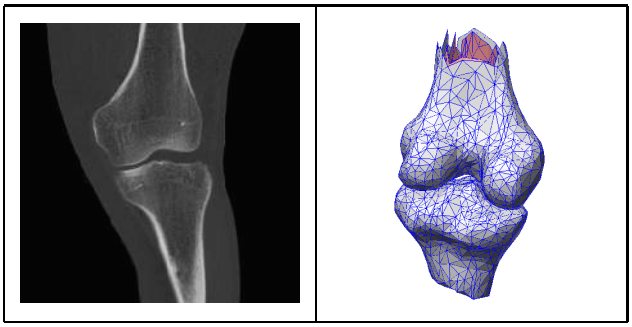


Fig 1.1: Left: CT scan of a human knee. Right: A pair of extracted C^0 iso-surfaces for the knee skeleton boundaries.

1.2 Prior Solution Approaches

Traditionally, a thin shell was often treated as a single surface (see [6, 25]) 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. Fig. 4.3 and Fig. 5.2). 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. [16, 24, 30]) of parametric surface splines to construct individual boundary surfaces as well as mid-surfaces of the fat boundaries. However, the independent construction of each surface not only increases tremendously the space and the time costs, but also fails to guarantee that these surfaces are always separate. Particularly, 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 [1], we proposed an adaptive approach in which the fat surface is defined by the contours of a single trivariate function F. The function is defined on a collection of triangular prisms (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.

1.3 Our Approach

In this paper, we considerably extend the reconstruction method of [1] to achieve (a) hierarchical Hh-multiresolution, (b) ϵ-bounded approximations and (c) the ability to capture sharp curve creases while being C^1 smooth ev-

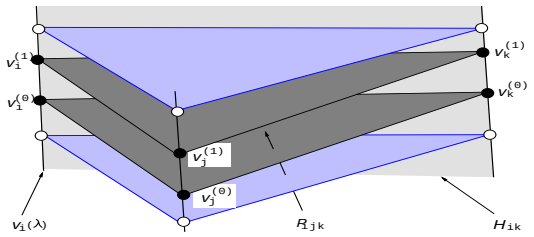


Fig 1.2: The volume prism cell P_{ijk}, the face $H_{ik}(t,\lambda)$ and the edge $v_i(\lambda)$ defined by a fat triangle $[V_iV_jV_k]$

erywhere 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.

1.4 Notation

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

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

For each fat triangle $[V_iV_jV_k]$, we further assume

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

Our trivariate function F for constructing the shell is piecewise defined on a collection of prisms. Let $[V_iV_jV_k]$ be a fat triangle. Then the prism, denoted by P_{ijk}, for $[V_iV_jV_k]$ is a volume in $\mathbb{R}^3$ enclosed by the surfaces H_{ij}, H_{jk}, and H_{ki} (see Fig. 1.2), where H_{lm} is a ruled surface defined by V_l and V_m as follows

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

with $v_i(\lambda) = V_i^{(0)} + \lambda N_i$, $N_i = V_i^{(1)} - V_i^{(0)}$. The prism P_{ijk} is a volume represented explicitly as

$$P_{ijk}(I) = \{p : p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda), \\ b_1 + b_2 + b_3 = 1, \quad b_l \geq 0, \quad \lambda \in I\},$$

where I is a specified interval. 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 $\lambda \in I$, $T_{ijk}(\lambda) := \{p : p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda), \quad b_1 + b_2 + b_3 = 1, b_l \geq 0\}$ 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 defined and computed in Appendix. We call the union of all $P_{ijk}(I_{ijk})$ a prism scaffold. For the input fat triangulation, 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 $[V_iV_j]$ and triangle $[V_iV_jV_k]$ 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 fat vertex 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 §3.

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. In the Appendix of this paper, we describe a method to compute the local coordinates from xyz coordinates.

The trivariate function defined on the shell is denoted by $\mathcal{F}$ in contrast with the function F for constructing the shell itself. Whenever it is necessary to address the functions that are defined on the level i scaffold (H-direction), the notations $F^{(i)}$ and $\mathcal{F}^{(i)}$ are used. Notations F_σ and $\mathcal{F}_\sigma$ will be used to address the level σ functions in the h-direction.

2 Algorithm Outline

This section gives the algorithm pipeline. The details of the algorithm are 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 fat triangulation. On this scaffold, a C^1 function $F^{(0)}$ is constructed (see §4.1). 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 [13, 17], with changes to the vertex removal criterion and hole re-triangulation method (see §3). Having such a DAG, we are able to travel from fine level to coarse 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 §3.3).

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 is 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 §4.3). Having these 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, $\sigma = 1, 2, \cdots, \Sigma$, so that

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

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 §4.4 and §5.

Step 6. *Evaluate and display the fat surface and any functions on them* See §6 and §7 for details.

3 Hierarchical Representation of Prism Scaffold

The hierarchical representation of the scaffold is a sequence $\mathcal{S}_0$, $\mathcal{S}_1$, $\cdots$, $\mathcal{S}_k$ of scaffolds, from the finest level to the coarsest. To construct the hierarchical representation of the scaffold, we perform a vertex removal procedure of the triangulation pair. The policy of the vertex removal is adopted from [13, 17]. That is, if one vertex is selected to be removed, then its neighbor vertices may not be removed. Hence any two vertices, in the set of vertices that are going to be removed, are disconnected (see Fig. 3.1). The next level of triangulation is obtained by re-triangulating holes that are left when the vertices are removed.

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 level i and $i+1$, there is an intermediate level which corresponds to the removed vertices of level i. There is an arc from the star-shaped polygon that is formed when a vertex is removed, to every triangle in level i around the vertex, and to every triangle in level $i+1$ formed by the re-triangulation of the star-shaped polygon. A polygon at the intermediate level that is between level i and level $i+1$ is called the *parent polygon* of prisms at level i if these prisms are linked to this polygon, and also it is called the *child polygon* of prisms at level $i+1$ if these prisms are linked to it. Unchanged triangles between two levels are linked directly by arcs. These descriptions are better illustrated by Fig. 3.1.

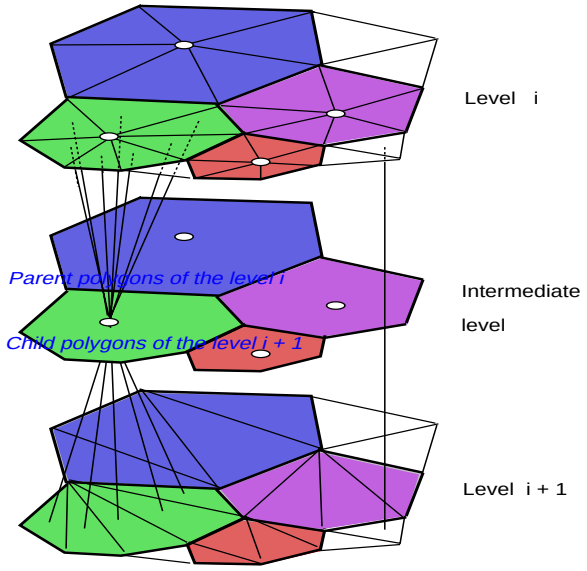


Fig 3.1: The vertices with circles on the top are the ones that are going to be removed at level i. The star-shaped polygons, that are shown in the middle of the figure, obtained by removing the selected vertices, are re-triangulated as shown in the bottom. Newly formed prisms at level $i+1$ are linked to the prisms at level i by arcs through the intermediate level. The unchanged prisms are linked directly.

Along with the DAG, we need to store some data. One is of course the fat vertex list *VertexList*. This list is fixed and does not change during the construction of the DAG. Another list is *FaceList*, that of the faces of the prism. This list 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 C^1 gradient on that face (see §4.3). When new prisms are produced, the new faces are added to this list. In the DAG, each prism needs to store three pointers that point to its three faces. The reason for having this information is to be able to construct C^1 functions later within each prism cell.

To achieve our goal of building the hierarchical representation of the scaffold, there are two points need to be addressed. One is the vertex removal criterion. The other is the re-triangulation. In the following two subsections, we detail these two steps.

3.1 Vertex Removal

The vertex removal 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 §4.2 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 fat vertices into two groups. One group contains the vertices that are not removable. The other group contains the remaining vertices. The unremovable group includes vertices that are marked as sharp (see §5), 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., [15, 18, 20, 21, 22, 32]). Any vertex removal scheme, for instance the scheme created by B. Hamann [20, 21] 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 be the set of removable vertices. Then for each vertex $v \in R$, we specify a grade to it that measures the normal variation. After all the vertices in R 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.

Let $p_i \in R$ and $T_{ijk},\ (i,j,k) \in K_i$ be all the triangles around p_i. Let $T_{jkl},\ (j,k,l) \in K_i'$ be the triangles formed by the re-triangulation after p_i is removed. Let G_{jkl} be the regrouping of the data set $p_i \cup (\cup_{(i,j,k)\in K_i} G_{ijk})$ into the prism $P_{jkl},\ (j,k,l) \in K_i'$, where $G_{ijk} = \emptyset$ at level zero. The sets $Gijk$ are recursively generated, with each being the regrouping of the union of similar sets at the previous level with a newly removed vertex. Then the grade for the vertex p_i is defined by

$$Grad(p_i) = \max_{(j,k,l)\in K_i'} \max_{p\in G_{jkl}} \max_{s=0,1} \max\{\theta_{jkl}^{(s)}(p), w\tilde{\theta}_{jkl}^{(s)}(p)\} \tag{3.1}$$

where $\theta_{jkl}^{(s)}(p)$ is the angle between the averaging normal $b_1^{(s)}N_j^{(s)} + b_2^{(s)}N_k^{(s)} + b_3^{(s)}N_l^{(s)}$ and normal $N^{(s)}$ of vertex p, and $\tilde{\theta}_{jkl}^{(s)}(p)$ is the angle between normal $N_{jkl}^{(s)}$ of the triangle $T_{jkl}^{(s)}$ and normal $N^{(s)}$. $N_{jkl}^{(s)}$ is assumed to point to the positive side of $T_{jkl}^{(s)}$. w is a weight. We take $w = 2$. This grade is attached 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 attached zero grade. We use the same notation $Grad(\cdot)$ to denote the grade of a prism.

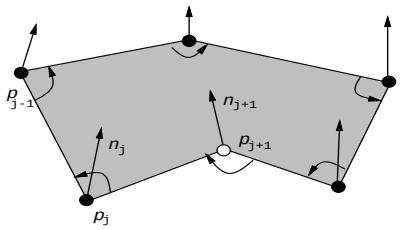


Fig 3.2: Convexity test by right-hand rule: The vertices with black and white dots are labeled as convex and non-convex, respectively.

3.2 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 fat triangulation. The other surface is triangulated in the same manner, preserving similar topology for the twin. Let $p_0^{(1)}, p_1^{(1)}, \cdots, p_n^{(1)}$ 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 $n \geq 3$, then label the point $p_j^{(1)}$ as convex if (see Fig. 3.2) $[(p_{j+1}^{(1)} - p_j^{(1)}) \times (p_{j-1}^{(1)} - p_j^{(1)})]^T n_j^{(1)} > 0$, label it as non-convex otherwise, where $\times$ denotes cross product.

(3). Find (j,k), $0 \leq j,k \leq n$, 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 triangulation $\{T_{ijk}\}$, $(i,j,k) \in K_i$ is minimal.

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

3.3 Adaptive Extraction of Fat Surface Support

It is obvious that simply taking a certain level of the scaffold from the hierarchy would not have the adaptive nature. Therefore, it is necessary to combine different level scaffolds to form an adaptive one. The extraction algorithm in [13, 17] 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 i. Then we denote by $G_i(P)$ the collection of all prisms in the level i that are in the same child polygon as P, and $G_i^c(P)$ the collection of all prisms in the level $i-1$ that are linked to child polygon of P. Let $Sub_i(P)$ be the collection of all prisms in the levels $i, i-1, \cdots, 0$, that are linked directly or indirectly through intermediate nodes to the prisms in $G_i^c(P)$. That is, $Sub_i(P)$ consists of the prisms in the sub-DAG starting with $G_i(P)$. Then the algorithm for extracting scaffold can be described as the following C language style pseudo-code:

```
Q_k = S_k;                    /* put all the prisms in S_k to Q_k*/
for (i = k;  i > 0;  i--) {
    Q_{i-1} = NULL;
    while (Q_i ≠ NULL ) {
      P = Q_i[0];
      if (G_i(P) ⊄ Q_i) {
          accept all the prisms in G_i(P);
      }  else {
          if (Grad(p) ≤ g for all p ∈ Sub_i(P) ) {
            accept all the prisms in G_i(P)
          } else {
            append to the end of Q_{i-1} all the prisms in
G_i^c(P)
          }
      }
      remove from Q_i all the prisms that in G_i(P);
    }
}
if (Q_0 ≠ NULL) {
    accept all the prisms in Q_0;
}
```

4 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 first constructed (§4.1). 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 (§4.3). (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 (§4.4).

4.1 Function over the Finest Level $\mathcal{S}_0$

The function $F^{(0)}$ 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 then the function is defined in 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 $[v_l(\lambda)\ v_m(\lambda)] = \{p \in I\!R^3 : p = H_{lm}(t, \lambda), t \in [0, 1]\}$ 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

$$\begin{aligned} F(H_{lm}(t,\lambda)) = & F(v_l(\lambda))H_0^3(t) + F(v_m(\lambda))H_2^3(t) \\ & + [v_m(\lambda) - v_l(\lambda)]^T \nabla F(v_l(\lambda)) H_1^3(t) \\ & + [v_m(\lambda) - v_l(\lambda)]^T \nabla F(v_m(\lambda)) H_3^3(t), \end{aligned} \tag{4.1}$$

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

$$F(v_i(\lambda)) = 2\lambda - 1,\ \nabla F(v_i(\lambda)) = (1-\lambda)N_i^{(0)} + \lambda N_i^{(1)}. \tag{4.2}$$

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

$$\begin{aligned} & d_1(\lambda) = v_m(\lambda) - v_l(\lambda), \\ & d_2(t) = (1-t)N_l + tN_m, \\ & d_3(t, \lambda) = d_1 \times d_2. \end{aligned} \tag{4.3}$$

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

$$\begin{cases} d_1^T \nabla F(H_{lm}(t,\lambda)) = \dfrac{\partial F(H_{lm}(t,\lambda))}{\partial t}, \\ d_2^T \nabla F(H_{lm}(t,\lambda)) = \dfrac{\partial F(H_{lm}(t,\lambda))}{\partial \lambda}, \\ d_3^T \nabla F(H_{lm}(t,\lambda)) = d_3^T \nabla \breve{F}_{lm}(t,\lambda), \end{cases} \tag{4.4}$$

where

$$\nabla \breve{F}_{lm}(t,\lambda) = (1-t)\nabla F(v_l(\lambda)) + t\nabla F(v_m(\lambda)) \tag{4.5}$$

From (4.4) we have

$$\nabla F(H_{lm}(t,\lambda))^T = [P, Q, R][d_1, d_2, d_3]^{-1} \tag{4.6}$$

where $[d_1, d_2, d_3]^{-1} = [d_1 \|d_2\|^2 - d_2(d_1^T d_2),\ d_2\|d_1\|^2 - d_1(d_1^T d_2),\ d_3]^T / \|d_3\|^2$, and P, Q and R are the right-hand sides of (4.4).

Next we define C^1 functions within prisms. Let $[V_1V_2V_3]$ be a typical fat triangle. The C^1 function F

in the prism P_{123} is defined by the side-vertex scheme defined by Theorem 3.1 in [29]:

$$F(p_{123}(b_1,b_2,b_3,\lambda)) = \sum_{i=1}^{3} w_i D_i(b_1,b_2,b_3,\lambda) \qquad (4.7)$$

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 [1] for detail).

4.2 Minimal Prism with ϵ Offset

As we have pointed out that the shell $\{p \in I\!R^3 : F^{(0)}(p) \in [-1,1]\}$ constructed in this section will serve as exact, 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 a triangular shell with ϵ offset 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 $[V_iV_jV_k]$ be a fat triangle. Let $p^{(0)}$ be the point in $I\!R^3$ 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

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

Then we require $|\lambda^{(0)} - \lambda^{(1)}| \geq \epsilon/\sqrt{M}$, where $M := M(N_i, N_j, N_k)$ is the minimal value of the degree two Bézier polynomial $\|b_1N_i + b_2N_j + b_3N_k\|^2$ on the triangle $\{b_1 + b_2 + b_3 = 1, b_i \geq 0\}$. 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 $P_{ijk}(I_{ijk}^{\epsilon})$ with $I_{ijk}^{\epsilon} = [a - \epsilon/\sqrt{M}, b + \epsilon/\sqrt{M}]$. The interval $[a,b]$ can be computed by numerical methods (see §6).

4.3 Compute Face Data

The function F in each prism is defined by transfinite interpolation of the data on the face of the prism (see §4.1 or §4.4). To have $F \ C^1$ 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 every prism in every level. For the finest level, these functions have been defined by (4.1) and (4.4). Now we consider the functions on other levels. Though the face data on level $i+1$ could be incrementally computed from the data of level i, we compute data on level $i+1$ 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

$$F(H_{lm}(t,\lambda)) = G_{lm}(t,\lambda) + \phi_{lm}^{\sigma}(t) + \psi_{lm}^{\sigma}(t)\lambda \qquad (4.8)$$

where $G_{lm}(t,\lambda)$ takes the same form as $F(H_{lm})$ in (4.1), 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 $t_i = i/2^\sigma$, $i = 0, 1, \cdots, 2^\sigma$. Here we shift N_{i3}^{σ} so that t_i is the center of the support $\mathrm{supp}N_{i3}^{\sigma} = ((i-2)/2^\sigma, (i+2)/2^\sigma)$. 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:

$$\int_0^1 \left[F(H_{lm}(t,\lambda_s(t))) + (-1)^s\right]^2 \, dt = min, \quad s = 0,1 \qquad (4.9)$$

where $\lambda_s(t)$, for fixed t, is defined by the intersection point of the line $(1-t)v_l(\lambda) + tv_m(\lambda)$ with the surface $F^{(0)} + (-1)^s = 0$. The required pieces of the intersection are obtained from the DAG. The minimization in (4.9) leads to a system of linear equations

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

with $c_j = -\int_0^1 \left[G_{lm}(t,\lambda_s(t)) + (-1)^s\right] N_{j3}^{\sigma}(t)\, dt$ and $j = 2, \cdots, 2^\sigma - 2, \quad s = 0,1$. 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. 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.

Then we define the gradient $\nabla F(H_{lm}(t,\lambda))$ by the conditions (4.4), but $\breve{F}_{lm}(t,\lambda)$ is modified by adding a spline function:

$$\nabla\breve{F}_{lm}(t,\lambda) = (1-t)\nabla F(v_l(\lambda)) + t\nabla F(v_m(\lambda)) + \breve{\phi}_{lm}^{\sigma}(t) + \breve{\psi}_{lm}^{\sigma}(t)\lambda \qquad (4.10)$$

with $\breve{\phi}^\sigma_{lm}(t) = \sum_{i=2}^{2^\sigma-2} \breve{\phi}_i N^\sigma_{i3}(t)$, $\breve{\psi}^\sigma_{lm}(t) = \sum_{i=2}^{2^\sigma-2} \breve{\psi}_i N^\sigma_{i3}(t)$, where $\breve{\phi}_i, \breve{\psi}_i \in I\!R^3$ are determined by

$$\int_0^1 \|\nabla \breve{F}_{lm}(t, \lambda_s(t)) - \nabla F^{(0)}(H_{lm}(t, \lambda_s(t)))\|^2 dt = min \tag{4.11}$$

for $s = 0, 1$, and $\lambda_s(t)$ is defined as before. (4.11) can be solved together with (4.9), since they share the same coefficient matrix.

4.4 Construction of C^1 Spline Approximations

In this section, we construct a piecewise C^1 function $F = F_\sigma$ ($\sigma \geq 0$ 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 with a regular partition, C^1 cubic splines defined in BB form were given by Sabin, 1976 (see [31]). Fig. 4.1 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 Bőhm, Farin and Kahmann [7]). However, the collection we use is indeed linearly independent. For a regular partition of a triangle, say 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 $(i, j, k) \in J^\sigma = J^\sigma_1 \cup J^\sigma_2$, where J^σ_1 and J^σ_2 are defined as follows:

$$\begin{aligned} J^\sigma_1 = \{(i,j,k): \quad & i,j,k \in \{1,2,3,...,2^\sigma\}; \\ & i+j+k = 2^\sigma + 2\}, \\ J^\sigma_2 = \{(i,j,k): \quad & i,j,k \in \{1,2,3,...,2^\sigma - 1\}; \\ & i+j+k = 2^\sigma + 1\}, \end{aligned}$$

where 2^σ is the resolution of the partition. Fig. 4.2 gives J_1 and J_2 for $\sigma = 2$. Now we denote the base function defined by Fig. 4.1 with center triangle T_{ijk} as N^σ_{ijk}.

4.4.1 F on Prisms

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

$$F_\sigma(p_{123}(b_1,b_2,b_3,\lambda)) = \sum_{i=1}^{3} w_i D_i(b_1,b_2,b_3,\lambda) + T_\sigma(b_1,b_2,b_3,\lambda) \tag{4.12}$$

where the first term of left-hand side is in the same form as (4.7), and the second term is a spline function:

$$T_\sigma(b_1,b_2,b_3,\lambda) = \sum_{(i,j,k)\in J^\sigma_3} (a_{ijk} + w_{ijk}\lambda) N^\sigma_{ijk}(b_1,b_2,b_3),$$

with $J^\sigma_3 = \{(i,j,k) \in J^\sigma : i > 1, j > 1, k > 1\}$. This is called a *correction term*, which is used to fit the finest

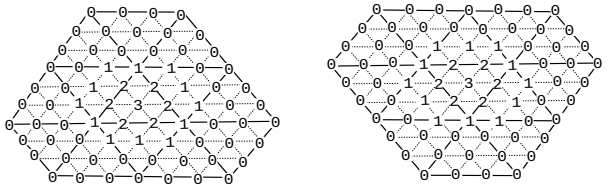


Fig 4.1: Bézier coefficients for two C^1 cubic spline basis functions. Each is defined on the union of 13 sub-triangles, which forms the support of the function.

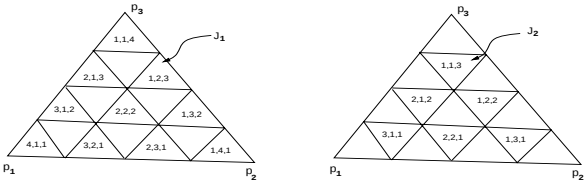


Fig 4.2: For the regular partition of a triangle with resolution 2^σ, the index set J^σ of the sub-triangles is divided into J^σ_1 and J^σ_2. This figure shows them for $\sigma = 2$.

level fat surface in the least square sense:

$$\iint_\Delta [F_\sigma(b_1,b_2,b_3,\lambda_s(b_1,b_2,b_3)) - (-1)^s]^2 dS = min \tag{4.13}$$

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 $\{(b_1,b_2,b_3) : b_1+b_2+b_3 = 1, b_i \geq 0\}$. The minimization in (4.13) leads to a system of linear equations.

4.4.2 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 §4.4.1 is not incremental. 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 §4.4.1, so that F is progressively constructed. Now we want to have the following form expression:

$$T_\sigma = T_{\sigma-1} + \sum_{(i,j,k)\in J^\sigma_3 \setminus 2*J^{\sigma-1}_3} (a^\sigma_{ijk} + w^\sigma_{ijk}\lambda) N^\sigma_{ijk},$$

where $T_1 = 0$. Let

$$W^\tau = \text{span}\{N^\tau_{ijk} : \ i \in J^\tau_3 \setminus 2 * J^{\tau-1}_3\},$$

$$S^\tau = W^2 \oplus W^3 \oplus \cdots \oplus W^\tau.$$

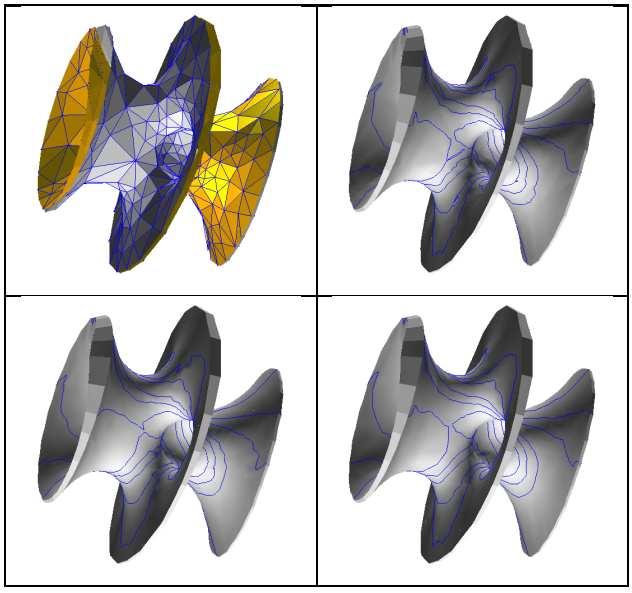


Fig 4.3: h-direction hierarchical multiresolution construction for the hypersheet surface triangulation (513 fat triangles) of level 2 of the DAG with $\sigma = 1$(top-right), $\sigma = 2$(bottom-left) and $\sigma = 3$(bottom-right). The level 0 triangulation has 917 fat triangles with varying thickness.

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^σ_{ijk} and w^σ_{ijk} 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 could 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^σ_{ijk} be the fitting error. Then for any given fitting error tolerance ϵ, we can choose a minimal σ so that $\epsilon^\sigma_{ijk} \le \epsilon$. This σ is prism dependent.

Basic Result *The composite function F, defined on any extracted scaffold and for any varying and prism-dependent $\sigma \ge 0$, 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. 4.3 provides examples of hierarchical multiresolution construction in the h-direction. The figures (for $\sigma = 1, 2, 3$) 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.

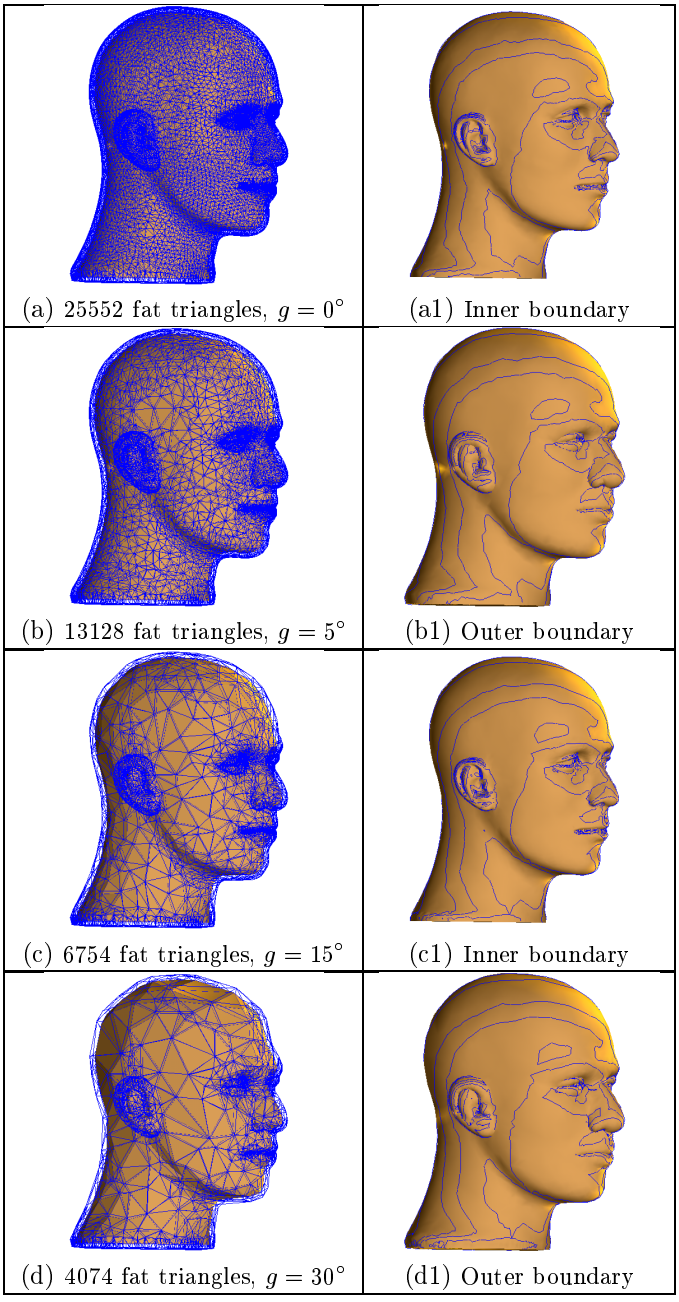
The H-direction multiresolution is shown in Fig. 4.4. To see the pairwise nature of the triangulation, $\mathcal{T}^{(1)}$ is plotted by wire, while $\mathcal{T}^{(0)}$ is plotted by piecewise planes. Fig. (a) is the original triangulation pair that has 25561 fat triangles. Figs. (b), (c) and (d), which have 13141, 6765 and 4069 fat triangles respectively, are the extracted triangulation. More examples are given in Fig. 8.1.

5 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 triangulation $T^{(0)}$ and $T^{(1)}$. If $\theta > \alpha$, then this edge is marked as a sharp edge. Here θ_1 is the angle between the two normals of the two triangles and α is a threshold value for 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. Now the triangles around a sharp vertex are divided into some groups by the sharp edges (see Figure 5.1). For each group, we assign a single normal for the vertex. This normal can 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 (4.8) and the gradient $\nabla \breve{F}_{lm}$ defined in (4.10) need to be redefined. The function $F(H_{lm}(t, \lambda))$ is changed to be the averaging function of the two local fitting functions that are defined on the neighbor volumes of the face. If there is only one neighbor volume, for a boundary edge, the face function is taken to be the volume function on that face. The averaging 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 (4.11) 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. 5.2. 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 fat 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



(a) 25552 fat triangles, $g = 0^\circ$ (a1) Inner boundary

(b) 13128 fat triangles, $g = 5^\circ$ (b1) Outer boundary

(c) 6754 fat triangles, $g = 15^\circ$ (c1) Inner boundary

(d) 4074 fat triangles, $g = 30^\circ$ (d1) Outer boundary

Fig 4.4: H-direction smooth reconstruction of fat triangulations: (a) the input triangulation. (b), (c) and (d) are adaptively extracted meshes from the DAG with $g = 5^\circ, 15^\circ, 30^\circ$, respectively. The right column shows inner/outer boundary surfaces with isophotes showing the smoothness. The level in the h-direction is zero.

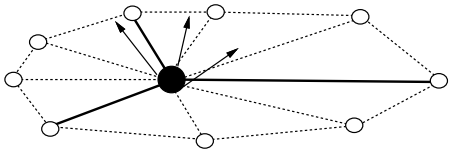


Fig 5.1: Grouping the triangles by the sharp edges (thick lines) and assigning a normal for each group.

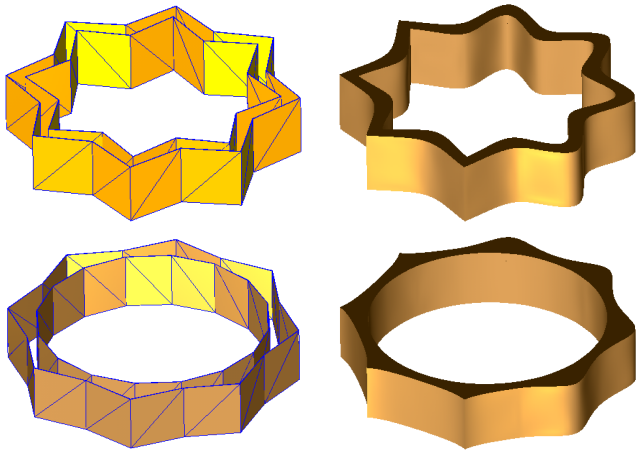


Fig 5.2: Left: the input polygons with some edges marked as sharp. Right: the constructed fat surfaces with sharp curve creases. There are four fat edges (inner and outer) on the top polygon marked as sharp. On the bottom polygon, only four outer edges are marked.

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).

6 Evaluation and Display of Fat 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 fat triangle. Then for each (b_1, b_2, b_3), $b_i \geq 0$, $\sum b_i = 1$, determine $\lambda_{min}^{(\alpha)} = \lambda_{min}^{(\alpha)}(b_1, b_2, b_3)$ such that

$$F(p_{ijk}(b_1, b_2, b_3, \lambda_{min}^{(\alpha)})) = \alpha,$$
$$\left|\lambda_{min}^{(\alpha)} - \tfrac{1}{2}\right| = \min\left\{\left|\lambda - \tfrac{1}{2}\right| : \ F(p_{ijk}(b_1, b_2, b_3, \lambda)) = \alpha\right\}. \tag{6.1}$$

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 (4.7) that $D_i(b_1, b_2, b_3, \lambda)$ is a rational function of λ. It is in the form

$$f_0 + f_1\lambda + f_2\lambda^2 + \frac{N_0 + N_1\lambda + N_2\lambda^2 + N_3\lambda^3 + N_4\lambda^4}{D_0 + D_1\lambda + D_2\lambda^2}. \tag{6.2}$$

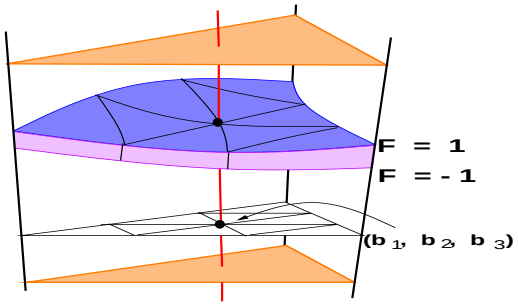


Fig 6.1: Evaluation of a fat surface: The surface is parameterized in each prism with the triangle $\{b_1+b_2+b_3=1,\ b_i \geq 0\}$ as its domain.

Hence $\phi(\lambda) := F(p_{ijk}(b_1,b_2,b_3,\lambda))$ is a rational function of λ of the form (6.2). 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 (6.2) 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. 6.2), an onion-like structure.

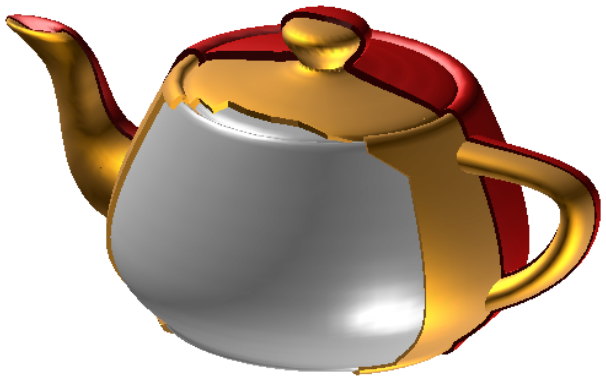


Fig 6.2: Middle surface and multiple layers of a shell: Inner surface and two layers are presented.

Using the hierarchical representation of the correction term, the evaluation of the fat surface can be progressive in the h-direction. Since $F_\sigma - F_{\sigma+1}$ in general is small, $F_\sigma=\alpha$ is 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 (6.2) 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.

7 Construction and Display of Functions on Fat Surfaces

The construction process of F is valid for constructing functions on the shell. In this section, we address the differences. Without loss of generality, we assume there is only one function value on each vertex, since each function could be constructed independently.

a. Vertex data. Instead of interpolating a pair of function values $\{-1,1\}$ at a fat vertex in the shell construction, we interpolate another pair of function values $\{f_i^{(0)}, f_i^{(1)}\}$. The normal pair $\{N_i^{(0)}, N_i^{(1)}\}$ is replaced by a gradient pair $\{g_i^{(0)}, g_i^{(1)}\}$. These gradients can be computed by locally fitting the function values by a quadratic, similar to the normal estimation procedure (see the Appendix). Now the function on the edge $v_i(\lambda)$, that is defined by (4.2) for the shell of a prism, is replaced by

$$\begin{aligned}\mathcal{F}(v_i(\lambda)) = & f_i^{(0)} H_0^3(\lambda) + N_i^T g_i^{(0)} H_1^3(\lambda) \\ & + f_i^{(1)} H_2^3(\lambda) + N_i^T g_i^{(1)} H_3^3(\lambda).\end{aligned}$$

The gradient on the edge $v_i(\lambda)$ is replaced by

$$\nabla\mathcal{F}(v_i(\lambda)) = (1-\lambda)g_i^{(0)} + \lambda g_i^{(1)} + \lambda(1-\lambda)B(\lambda)$$

where $B(\lambda)$ is defined by the conditions $d\mathcal{F}(v_i(\lambda))/d\lambda = N_i^T \nabla\mathcal{F}(v_i(\lambda))$ and $\|B(\lambda)\| = min$. This yields

$$B(\lambda) = 3N_i\{2[f_i^{(1)} - f_i^{(0)}] - N_l^T[g_i^{(0)} + g_i^{(1)}]\}/\|N_i\|^2.$$

b. Face data. In determining ϕ_i and ψ_i in (4.8), equation (4.9) is replaced by

$$\int_0^1 \left[\mathcal{F}(H_{lm}(t,\lambda_s(t))) - \mathcal{F}^{(0)}(H_{lm}(t,\lambda_s(t)))\right]^2 dt = min$$

for $s=0,1$, where $\mathcal{F}^{(0)}$ is the function defined on $\mathcal{S}_0$.

c. Function in prisms. In determining a_{ijk} and w_{ijk} in (4.12) for the prism P_{ijk}, equation (4.13) is replaced by

$$\begin{aligned}\iint_\Delta & [\mathcal{F}_\sigma(p_{123}(b_1,b_2,b_3,\lambda_s(b_1,b_2,b_3))) - \\ & \mathcal{F}^{(0)}(p_{123}(b_1,b_2,b_3,\lambda_s(b_1,b_2,b_3)))]^2\ dS = min\end{aligned}$$

for $s=0,1$.

d. Adaptive Construction of $\mathcal{F}$. To have the construction of $\mathcal{F}$ adaptive as well, the grade defined in (3.1) needs to combine one more term $\vartheta^{(s)}_{jkl}(p)$ that is necessary for $\mathcal{F}$. This term is the same as $\theta^{(s)}_{jkl}(p)$ but with changing normals to gradients in its definition.

e. Evaluation of $\mathcal{F}$. Evaluation of $\mathcal{F}$ is straightforward. It should go along with the evaluation of the shell. For every computed (b_1, b_2, b_3, λ) for the shell in a prism, simply compute $\mathcal{F}$ to get the function value.

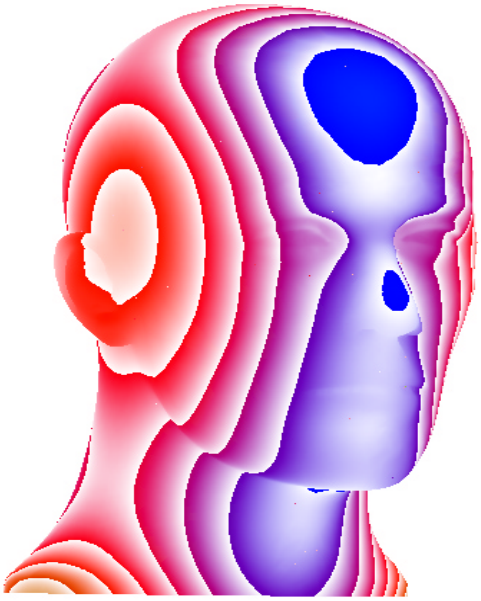


Fig 7.1: Contour plot of an acoustic pressure function defined on the head. Smooth iso-contours show that both the shell and the function on the shell are smooth.

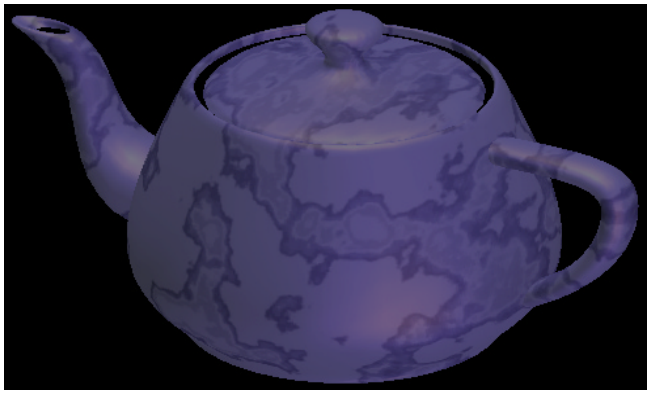


Fig 7.2: Texture map on the inner surface of Fig. 6.2. A marble 3D texture function is used.

Fig. 7.1 shows the contours for a function defined on the head with different colors in the region between two contours. The function value represents the pressure of the acoustics on the head. The data is obtained from finite element analysis of the acoustic scattering problem—Helmholtz differential equation. Fig. 7.2 shows texture mapping on the teapot. The 3D texture $I(u, v, w)$ represents a piece of marble.

8 Conclusions

We have presented an adaptive hierarchical Hh-multiresolution reconstruction algorithm to model smooth fat surface objects from a matched triangulation pair. We also allow function values to be attached to the input vertices of the triangulations, so that physical data fields defined on the shell can be visualized or texture mapping can be performed. The implementation and test show that the proposed method is correct and fulfills our purpose.

References

[1] ——– *Smooth Adaptive Reconstruction and Deformation of Free-Form Fat Surfaces.* Technical report, Submitted, 1999.

[2] N. Amenta, M. Bern, and M. Kamvysselis. A New Voronoi-Based Surface Reconstruction Algorithm. In *Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH98*, pages 415–421, 1999.

[3] D. Attali. R-regular shape reconstruction from unorganized points. *Computational Geometry: Theory and Applications*, 10(4):239–247, 1998.

[4] C. Bajaj, F. Bernardini, and G. Xu. Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. In *Computer Graphics Proceedings, Annual Conference series, Proceedings of SIGGRAPH95, ACM SIGGRAPH95*, pages 109–118, 1995.

[5] R. E. Barnhill, K. Opitz, and H. Pottmann. Fat surfaces: a trivariate approach to triangle-based interpolation on surfaces. *Computer Aided Geometric Design*, 9:365–378, 1992.

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

[7] W. Bőhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[8] M. L. Bucalem and K. J. Bathe. Finite element analysis of shell structures. *Archives Comput. Methods Engrg.*, 4:3–61, 1997.

[9] B. K. Choi, H. Y. Shin, Y.I. Yoon, and R. Ramaraj. Triangulation of scattered data in 3D space. *Computed-Aided Design*, 20:239–248, 1988.

[10] F. Cirak, M. Ortiz, and P. Schroder. *Subdivision Surfaces: A New Paradigm for Thin-Shell Finite-Element Analysis.* Technical report, http://www.multires.caltech.edu/pubs/, 1999.

[11] J. Cohen, A. Varshney, D. Manocha, G. Turk, and H. Weber. Simplification Envelopes. In *Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH96*, pages 119–128, 1996.

[12] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 303–312, 1996.

[13] M. de Berg and K. T. G. Dobrindt. On levels of detail in terrains. *GMIP*, 60:1–12, 1998.

[14] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, January 1994.

[15] C. Erikson. *Polygonal Simplification: An Overview.* Technical report, TR96-016, Department of Computer Science, UNC-Chapel Hill, 1996.

[16] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Second Edition.* Academic Press Inc., 1990.

[17] L. D. Floriani, P. Magillo, and E. Puppo. Data structures for simplicial multi-complexes. 1999.

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

[19] G. Golub and C. Van Loan. Matrix Computations. The Johns Hopkins University Press, 1983.

[20] B. Hamann. Curvature approximation for triangulated surfaces. In G. Farin, H. Hagen, and H. Noltemeier, editors, Geometric Modelling, Computing Suppl. 8, pages 139–153. Springer-Verlag, 1993.

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

[22] H. Hoppe, T. DeRose, T. Duchamp, M. Halstend, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. *Piecewise Smooth Surfaces Reconstruction.* In *Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH94*, pages 295–302, 1994.

[23] H. Hoppe, T. DeRose, T. Duchamp, J. M., and W. Stuetzle. Surface reconstruction from unorganized points. In Edwin E. Catmull, editor, *SIGGRAPH '92 Proceedings*, volume 26, pages 71–78, July 1992.

[24] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design.* A. K. Peters, 1993.

[25] M. S. Ketchum and M. A. Ketchum. *Types and Forms of Shell Structures.* 1997.

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

[27] H. Mollmann. *Introduction to the Theory of Thin Shells.* Chichester, New York, 1981.

[28] D. Moore and J. Warren. Approximation of dense scattered data using algebraic surfaces. In *Proc. of the 24th Hawaii Intl. Conference on System Sciences*, pages 681–690, Kauai, Hawaii, 1991.

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

[30] L. Piegl and W. Tiller. *The NURBS Book.* Springer, 1997.

[31] M. Sabin. *The use of piecewise form of numerical representation of shape.* PhD thesis, Hungarian Academy of Science, Budapest, 1976.

[32] G. Schroeder, J. A. Zarge, and W. E. Lorensen. *Decimation of Triangle Meshes.* In *Computer Graphics Proceedings, Annual Conference series, ACM SIGGRAPH92*, pages 65–70, 1992.

[33] B. Szabo and I. Babuska. *Finite Element Analysis.* New York : Wiley, 1991.

A Appendix

1. Compute P_{ijk}–Coordinate

For a given $p \in I\!R^3$, compute $(b_1, b_2, b_3, \lambda)^T$ such that

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

It follows from (A.1) that we have

$$p - v_k(\lambda) = [v_i(\lambda) - v_k(\lambda), v_j(\lambda) - v_k(\lambda)]\,[b_1,\ b_2]^T.$$

Therefore,

$$det[p - v_k(\lambda), v_i(\lambda) - v_k(\lambda), v_j(\lambda) - v_k(\lambda)] = 0. \quad \text{(A.2)}$$

The left-hand side of (A.2) is a polynomial of degree 3 in λ. Solve the equation for λ, then choose the root such that the solution (b_1, b_2, b_3) of (A.1) satisfies $b_i \geq 0$, $\sum b_i = 1$.

2. Normal Estimation and Pairing up the Vertices.

We estimate normals and pair up the vertex data as well by fitting the surface data. We use the weighted least square approximation to fit locally the data around each vertex. Let $V_i^{(0)} = (x_i^{(0)}, y_i^{(0)}, z_i^{(0)})^T \in D^{(0)}$ be a given vertex and $p_j^{(0)} \in D^{(0)}$, $j = 1, 2, \cdots, m$, and $p_j^{(1)} \in D^{(1)}$, $j = 1, 2, \cdots, n$, be some neighbor points of $V_i^{(0)}$. Let

$$Q_k(x, y, z) = -1 + \sum_{0<u+v+w\leq k} q_{uvw} \left(x - x_i^{(0)}\right)^u \left(y - y_i^{(0)}\right)^v \left(z - z_i^{(0)}\right)^w$$

be the fitting function, where the degree k is chosen to be 2 or 3 in our implementation. Then determine

the coefficients q_{uvw} by solving the following constraint minimization problem:

$$\begin{cases} \sum_{j=1}^{m} \omega_j^{(0)} [Q_k(p_j^{(0)}) + 1]^2 + \\ \sum_{j=1}^{n} \omega_j^{(1)} [Q_k(p_j^{(1)}) + 1 + \alpha]^2 = \min, \\ q_{100}^2 + q_{010}^2 + q_{001}^2 = 1, \end{cases} \tag{A.3}$$

where α is a parameter that needs be solved together with the other unknowns, and the weight $\omega_j^{(l)}$ is determined by the distance from $p_j^{(l)}$ to $V_i^{(0)}$. For example, take $\omega_j^{(l)} = 1/\|p_j^{(l)} - V_i^{(0)}\|$. Note that the form of Q_k makes $V_i^{(0)}$ lie on the surface $\{Q_k(x, y, z) = -1\}$, and the second condition of (A.3) guarantees that $V_i^{(0)}$ is a regular point of the surface since $\|\nabla Q(V_i^{(0)})\| = 1$. The first summation of the first equation of (A.3) forces the points $p_j^{(0)}$ to be on the surface $\{Q_k(x, y, z) = -1\}$. The second summation forces the points $p_j^{(1)}$ to lie on another surface $\{Q_k(x, y, z) = -1 - \alpha\}$. To solve problem (A.3), let $x_1 = [q_{100}, q_{010}, q_{001}]^T$, and x_2 be a vector consisting of all the other coefficients in (A.3) plus the parameter α. Then equation (A.3) can be expressed as a least square problem in matrix form:

$$M[x_2, x_1]^T = 0, \quad \|x_1\| = 1, \tag{A.4}$$

where M is an $m + n$ by $(k + 3)(k + 1)^2/6$ matrix. We require $m + n \geq (k + 3)(k + 1)^2/6$. If problem (A.4) has no unique solution, then increase the number of fitting points or decrease the degree of the fitting polynomial. To solve (A.4), let the QR decomposition of M be

$$M = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix},$$

where Q is an orthogonal matrix, R_{ii} are upper triangular for $i = 1, 2$, and R_{22} is a 3×3 matrix. Then problem (A.4) is equivalent to

$$\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = 0, \quad \|x_1\| = 1.$$

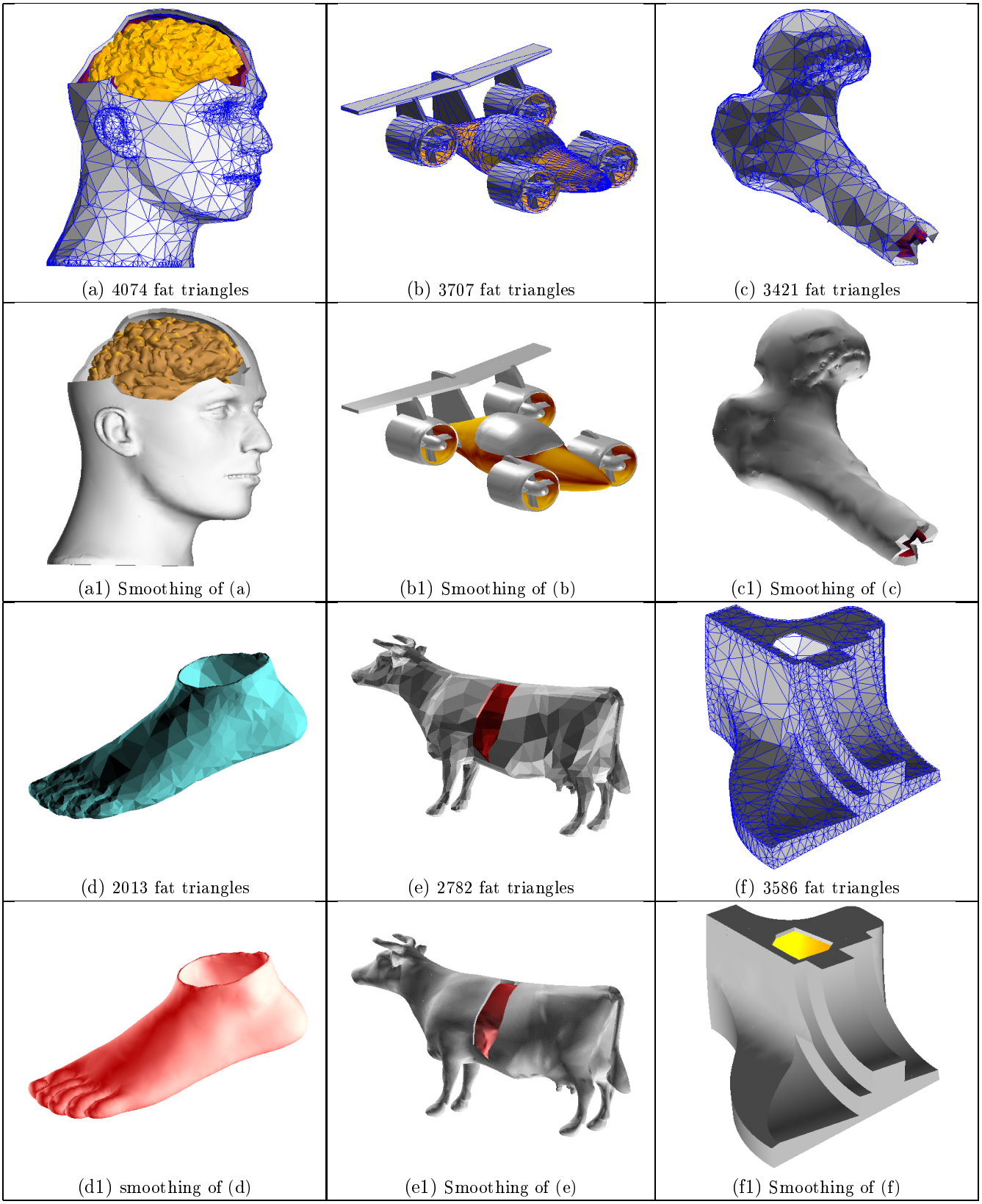
Using the singular value decomposition of $R_{22} = U \sum V^T$, we find that $x_1 = v_3$, the third column of V (see [19]), and then $x_2 = -R_{11}^{-1} R_{12} x_1$.

Having determined the fitting polynomial Q_k and the normal at $V_i^{(0)}$, which is x_1, the companion point $V_i^{(1)}$ of $V_i^{(0)}$ is computed as the intersection point of the surface $Q_k(x, y, z) = -1 - \alpha$ and the line $V_i^{(0)} + t x_1$. This leads to the following equation:

$$\sum_{s=2}^{k} t^s \left(\sum_{u+v+w=s} q_{uvw} q_{100}^u q_{010}^v q_{001}^w \right) + \alpha = 0.$$

Solve this equation for t and take the minimal solution in absolute value as the required t. Then take $V_i^{(1)} = V_i^{(0)} + t x_1$.

For any vertex $V_i^{(1)} \in D^{(1)}$, its companion point $V_i^{(0)}$ is similarly computed.



(a) 4074 fat triangles (b) 3707 fat triangles (c) 3421 fat triangles

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

(d) 2013 fat triangles (e) 2782 fat triangles (f) 3586 fat triangles

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

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